
mi computer
53
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
175 ptas.
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen V - Fascículo 53

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-007-4 (tomo 5)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 168501
Impreso en España - Printed in Spain - Enero 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.





Calidad de sonido



Marcus Wilson-Smith

La MIDI ofrece al usuario de un ordenador personal una forma de entrar en el extraño y fascinante mundo de la música electrónica

En el campo de la educación, la MIDI ofrece algunos verdaderos adelantos, tanto en la escuela como en el hogar. Aprender a leer música suele ser difícil, incluso cuando el estudiante toca ya bastante bien. Las primeras etapas en el aprendizaje de cosas tales como sostenidos y bemoles e indicaciones de compás pueden ser arduas y llevar mucho tiempo, y la relación entre la partitura y lo que en realidad se escucha muy raramente resulta obvia.

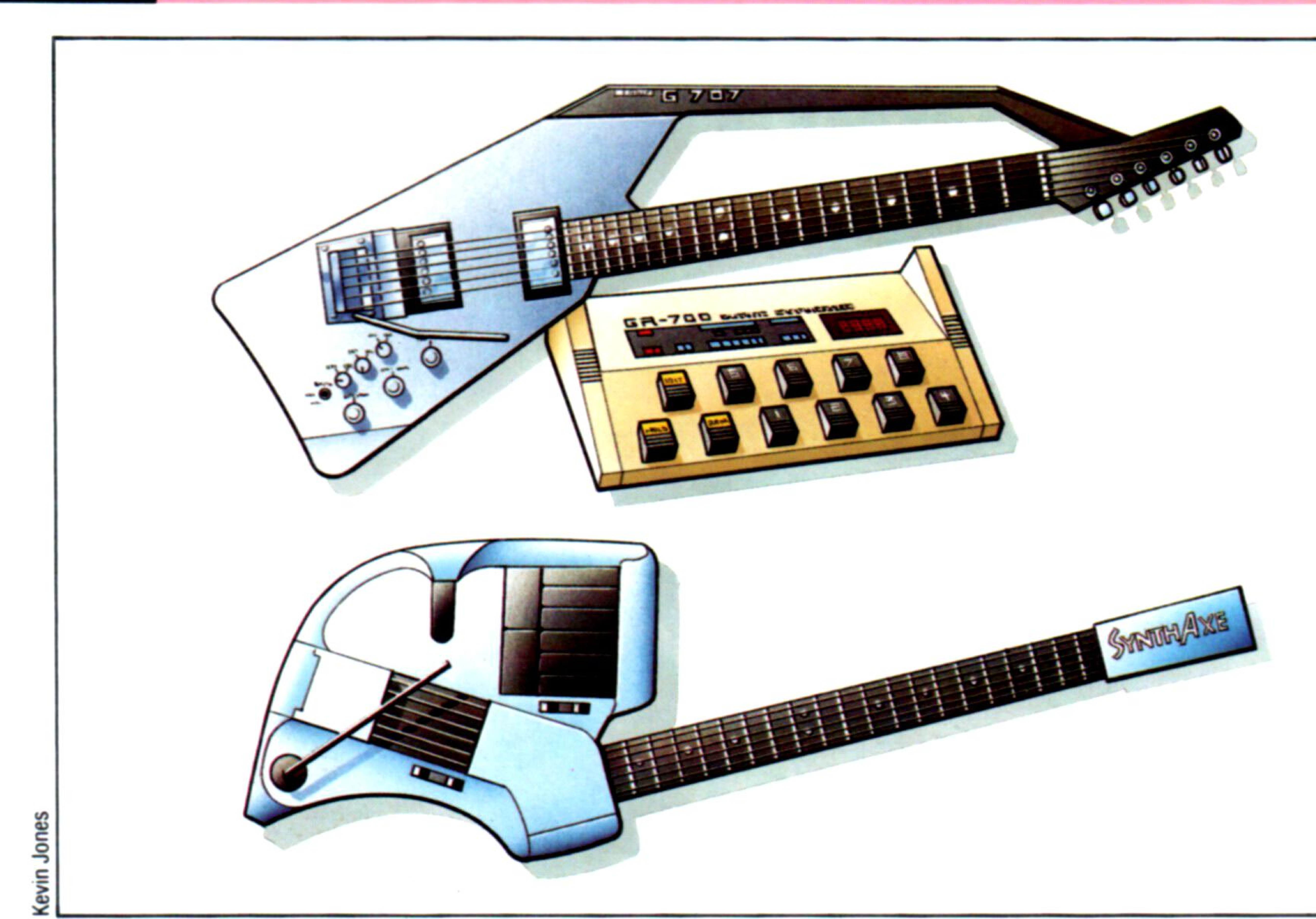
Uno de los principales aspectos del problema radica en la naturaleza de la música en sí: para que tenga algún sentido, deben coordinarse en el tiempo una serie de eventos. Si la única forma de que un principiante pueda seguir la notación gráfica es ir interrumpiendo la música, entonces todos los indicadores de duración de tiempo del pentagrama pierden todo sentido. De modo similar, el principiante suele tardar algún tiempo tratando de interpretar un compás o frase determinados; mientras realiza esto la música habrá pasado a ser otra cosa totalmente diferente.

La MIDI elimina estos obstáculos. Puede permitir almacenar la ejecución de un sintetizador en la memoria de un micro y, con el software apropiado, proporcionará una visualización gráfica de la música que se esté tocando. Ésta es una valiosa ayuda para cualquier estudiante de música. Significa que, por ejemplo, si se toca un *do* normal en el teclado del sintetizador, se mostrará un *do* normal en el pentagrama de la visualización en pantalla. Si se mantiene un acorde en *re menor* durante un cierto tiempo, se visualizarán los componentes armónicos del acorde (*re, fa, la*), junto con la duración adecuada.

Esta idea se puede ampliar mediante un software que ejecute una pieza de música preprogramada en el sintetizador conectado en interface mientras en la pantalla se va desplazando un pentagrama con la notación completa. En esta situación, tanto la música como su notación se pueden detener simultáneamente y volverse a ejecutar a partir de un número de compás especificado si el usuario tropieza con un problema. Además, se puede modificar el sonido global de la música cambiando los parámetros de control del sintetizador, introduciendo, por lo tanto, al usuario en el arte del arreglo.

Una vez adquirido un grado de confianza en cuanto a lectura de música, entonces será más fácil escribirla con la ayuda del teclado alfanumérico de un ordenador. Esto puede implicar la entrada de datos de ejecución sin ninguna referencia inmediata a un sonido del sintetizador, y comparar después el resultado con las intenciones originales. Una vez adquirida esta destreza más avanzada, se puede abandonar la notación en pentagrama en favor de otro sistema, como el MCL (*Music Composition Language:* lenguaje para composición musical). Para la música electrónica, el MCL es un medio más apropiado para entrar los datos, dado que incluye una especificación de características que son exclusivas de la producción de sonido electrónico. Para la aplicación MCL no se ha desarrollado ningún estándar: cada máquina posee su propio MCL. La notación de pentagrama, si bien constituye una guía visual inmediata, es un sistema cuya existencia data de varios siglos antes de la electrónica, y no se puede adaptar a todas las nuevas notaciones que requiere la música electrónica.

Sonidos nuevos, estilo nuevo
Uno de los numerosos nuevos sintetizadores de la gama DX de Yamaha, el DX7, incorpora un método para construir sonido, la síntesis FM, que previamente estaba restringido a máquinas muy caras. En vez de tomar un sonido existente y modificarlo haciéndolo pasar a través de filtros o adaptando los valores de envoltura, el DX7 crea sus propios sonidos complejos mediante la combinación de seis formas de onda de diversas maneras. Por consiguiente, el DX7 se aproxima a los sonidos de instrumentos acústicos mucho más que los otros sintetizadores. También incorpora control de respiración, de modo que un músico puede soplar en un receptor y agregarles a los sonidos del saxofón o la trompeta, por ejemplo, variaciones tipo respiración. El DX7 puede utilizar paquetes de RAM con características de sonido pregrabadas, o almacenar sonidos creados por el usuario en paquetes de RAM



Sonido sintetizado

Recientemente varios fabricantes han desarrollado sintetizadores que se activan desde las cuerdas de una guitarra en vez de desde las teclas de un piano. El Roland GR700 utiliza cuerdas de guitarra estándar. La compleja información del sonido se recoge mediante una entrada hexafónica (seis sonidos) y se envía al sintetizador, donde se le agregan parámetros ajenos a la guitarra. Un original enfoque a la misma técnica lo constituye el SynthAxe, un sintetizador controlado por 6809. El SynthAxe recoge datos de sonido desde una conexión eléctrica entre la cuerda y el traste. Unos sensores en la parte superior del clavijero captan las variaciones de inclinación y resbalamiento. Para mejorar aún más el sonido se utilizan un segundo grupo de cuerdas y un pequeño teclado

Kevin Jones

Para muchos usuarios de microordenadores, entender la notación de pentagrama o el MCL será la clave para aprovechar la MIDI al máximo. En cuanto a los estudiantes secundarios y universitarios, un obstáculo fundamental que deben salvar es el carácter que posee actualmente la educación musical. Incluso la mayoría de los estudiantes de música se ven abocados a un área de estudios firmemente asentada en la música clásica europea. La mayoría de los músicos clásicos y los profesores identifican la música electrónica con los compositores más vanguardistas o radicales de las dos últimas décadas y, hasta cierto punto, con la música *pop* contemporánea. Ninguno de estos campos se acepta en pie de igualdad con la música clásica. De hecho, algunos puristas llegan incluso a cuestionar que se trate siquiera de "música".

Parece improbable, por consiguiente, que el potencial de la MIDI en el área educativa se explore demasiado en la vertiente musical de la tendencia principal, especialmente porque, además de la electrónica musical, se requiere experiencia con ordenadores. Podrían existir algunos cursos de informática con facilidades para experimentación acústica que fueran adecuados para una clase de músicos de sintetizador MIDI. Si éste resultara ser popular en un medio de tales características, se requeriría una solución simple, tal como el empleo de auriculares. Pero, dado que la mayoría de las unidades MIDI están diseñadas para conectarse en interface con uno o más sintetizadores, para tratar con esto sería preciso hacer uso de varias señales. Incluso a un nivel básico, el empleo de la MIDI en educación implica el desarrollo de facilidades de música con ordenador en un estudio, y esto parece exigir una participación activa por parte de estudiantes de informática además de estudiantes de música.

Para las actuaciones en vivo, la MIDI es fundamentalmente un medio de integrar un cierto número de sintetizadores, secuenciadores y máquinas de ritmos en un único sistema controlable. La posibilidad que más preocupa a los músicos que utilizan una gran proporción de secuenciación en sus actuaciones es una pérdida de sincronización entre las unidades, y el resultante fracaso musical. Se ha sabido que músicos como los Thompson Twins y Howard Jones tocan cintas de apoyo grabadas en el estudio mientras "actúan en vivo", para no correr riesgos. Al menos en teoría los grupos de multisintetizadores deberían tener, gracias a la MIDI, actuaciones más libres de problemas.

Una característica de este avance es que tales grupos ya no tendrán apariencia de ser de "multisintetizadores". Desde principios de los años setenta, por lo general se ha pensado en el sintetizador como un instrumento de teclado, con gran número de mandos de control de parámetros y guías deslizadoras dispuestas en una tira encima de las teclas. Pero si un sintetizador de teclado está utilizando la MIDI para controlar a un segundo o un tercero, entonces ya no existe necesidad alguna de tener más teclados que el del instrumento maestro. A medida que el empleo de la MIDI se va generalizando, están saliendo al mercado más "módulos de sintetizador". Éstos son, simplemente, las unidades de generación de sonido y de secuenciación que previamente formaban parte de los instrumentos de teclado. Dado que estos módulos poseen poco o ningún interés *visual*, no hay necesidad de exhibirlos en el escenario.

Existe otro desarrollo que antecede a la MIDI, pero que es probable que sea objeto de más atención a medida que vaya disminuyendo la cantidad de teclados. Este desarrollo es la posibilidad de controlar la síntesis de sonido electrónico mediante datos de ejecución en cuerdas producidos con guitarras, y mediante datos de control de respiración y boca provenientes de instrumentos de viento. En comparación con la simple acción mecánica de pulsar una nota en un teclado, puntear una cuerda o hacer vibrar una lengüeta implica que al instrumento en cuestión se le está transmitiendo una información acústica más compleja. Si esta información se codifica digitalmente y se transmite



vía MIDI a un módulo de sintetizador fuera del escenario, no existe ninguna razón por la cual un saxofonista no pueda ser el principal controlador de la electrónica de un grupo, incluso de su máquina de ritmos. Yamaha ha incorporado control de respiración en su sintetizador DX7, y se ha diseñado la SynthAxe (una guitarra digital desarrollada recientemente) para emplear la MIDI con la misión de controlar la salida de un Fairlight.

Ello significa que se podría producir en el Yamaha un sonido de cuerda con las características de ejecución exclusivas de un saxofón y, de igual modo, se podría articular una muestra de trombón Fairlight mediante el rasgueado de una guitarra. Si bien ninguno de estos desarrollos es inminente (al fin y al cabo, el SynthAxe es una "guitarra" muy cara), constituyen indicios de probables tendencias para las actuaciones en vivo del futuro cercano. Es probable que se produzca una disminución gradual de los teclados en escena; adquirirán mayor importancia los instrumentos de cuerdas, viento y posiblemente de percusión afinada, como vibráfonos, y en la medida en que la tecnología del muestreo de sonidos se vuelva más asequible al usuario, podría llegar a predominar el sonido acústico.

Para finales de los ochenta, los tradicionalistas de la música puede que se sientan desagraviados al ver nuevamente una unidad estilo jazz de guitarra, saxofón, contrabajo y batería. Puede que, sin embargo, se sientan confundidos por el hecho de que el guitarrista esté tocando un vibráfono invisible y el saxo esté emitiendo sonidos de batería.

Para muchos grupos habituados a las actuaciones en vivo, la primera experiencia en un estudio de grabación avanzado puede ser intimidante. Se hallan frente a instrumentos musicales y sistemas operativos con los que nunca se habían encontrado antes y se les asigna un productor que tal vez no conozca su trabajo ni sus intenciones con particular profundidad. Y su compañía discográfica espera que produzcan versiones "mejores y más espectaculares" de sus éxitos en el escenario en este entorno tan poco familiar. El hecho se puede comparar con una compañía teatral semiprofesional a la que se colocara en el plató de una película de enorme presupuesto esperando un éxito de taquilla instantáneo. En algunas ocasiones la transición tiene éxito y todos los implicados quedan satisfechos. Pero con mucha frecuencia las ideas originales se pierden en un laberinto de dispositivos de estudio y al final esta magna empresa se ve abocada a un oneroso fracaso. Lo más frecuente es que éste se produzca cuando el grupo descarta su propio equipo, ya familiar (y, en realidad, su propio "sonido"), y se vuelca en los más deseables instrumentos que hay disponibles en el estudio. Pero puede que una idea que funcionaba bien en un sintetizador MiniMoog se malogre por completo al ejecutarse en un ordenador de muestreo digital Fairlight, y si este decepcionante resultado se produce con suficiente frecuencia, entonces la justificación para el empleo de tal estudio comienza a perder fuerza.

Sin embargo, si los músicos del grupo ya conocen la MIDI y si han utilizado un microordenador para almacenar secuencias y otros datos de control musical, ya estarán familiarizados con los procedimientos que se siguen en los estudios avanzados. En el nivel más inmediato, sería posible probar ideas empleando una sucesión de distintos instrumentos del estudio con un mínimo de dificultad, y con el conocimiento previo de que las ideas y las frases se pueden transformar simplemente mediante el cambio de sintetizadores.

El conocimiento de la MIDI también resulta valioso cuando se trata de familiarizarse con sistemas de estudio distintos de aquellos directamente implicados en la generación de sonido. Una mesa mezcladora de lógica transistorizada, por ejemplo, posee un ordenador exclusivo que recordará y volverá a ejecutar cualquier serie de decisiones efectuadas en las etapas finales de la grabación, lo que se conoce como mezclado. Cuando se ha grabado toda la música en 24 pistas de cinta separadas (la guitarra en una pista, las vocales de apoyo en otra, las vocales principales en otras tres, etc.), empieza la crucial tarea de equilibrar y mezclar todos los elementos. Generalmente es en este punto donde las partes individuales son tratadas con cualquier "efecto" requerido para hacer que destaquen o se fundan en la mezcla. Una única nota de trompeta puede necesitar el agregado de reverberación en un solo punto y, con otras 23 cosas sucediendo al mismo tiempo, es fácil equivocarse. Utilizar un ordenador para tratar y controlar tales incidentes durante el mezclado es como la secuenciación MIDI a gran escala.

Otra técnica, desarrollada originalmente para la sincronización y edición de video, pero que está adquiriendo protagonismo en la producción de música, es el empleo de código de tiempo. El código de tiempo es como un reloj digital más una señal de disparo, pero grabado en cinta. Utiliza palabras de 80 bits para proporcionar datos de sincronización cuando se graba música con secuencias de video y permite secuenciar juntos eventos musicales y montajes de video de fracciones de segundo.

Los músicos, en consecuencia, tienen fundadas razones para adquirir instrumentos compatibles con la MIDI; pero, además, ésta es una buena introducción a los sistemas musicales más avanzados.



Creatividad a gran escala
Laurie Anderson, poetisa y artista neoyorquina, realiza una insólita mezcla de sonidos y equipo de sonido con tecnología de cine, de cinta y de video, para crear un estilo exclusivo. En canciones como *O Superman* y *Mr Heartbreak* utiliza o es secundada por los más disímiles objetos, desde campanas africanas a instrumentos electrónicos ultramodernos como el Vocoder y el Synclavier

Método eficaz

En este capítulo estudiaremos la utilización de procedimientos para definir otros procedimientos

A fin de ilustrar este principio de emplear procedimientos para definir datos, tomemos como ejemplo el puzzle tangram. Se trata de un cuadrado que ha sido dividido en siete trozos geométricos, que se combinan de diversas maneras para formar distintas formas. En nuestro ejemplo utilizaremos las siete piezas básicas para crear una forma que se parezca a un perro. Comenzamos por definir procedimientos LOGO para cada una de las piezas básicas; estos "procedimientos de piezas" se incorporan luego a otro procedimiento, al que se le da el nombre de PERRO. Dado que la tortuga debe estar correctamente posicionada antes de que se dibuje cada pieza, también se deben utilizar otros procedimientos (desde MOVER1 a MOVER7).

Sería igualmente sencillo realizar este dibujo simplemente encadenando las instrucciones una tras otra en un procedimiento largo. Nuestro método utiliza los principios del diseño "de arriba abajo", o *top-down* (véase p. 956). En líneas muy generales, este método significa simplemente descomponer un problema en un número de partes y después proceder a resolver cada una de ellas individualmente. La gran ventaja de este enfoque es que el programador de LOGO puede definir un procedimiento que contenga subprocedimientos que aún no se han definido. El procedimiento principal no se puede ejecutar, por supuesto, hasta que se escriban los subprocedimientos o éstos sean sustituidos por rutinas ficticias. Para mostrarle cómo funciona esto, consideremos cómo se construyó el programa que dibuja al perro.

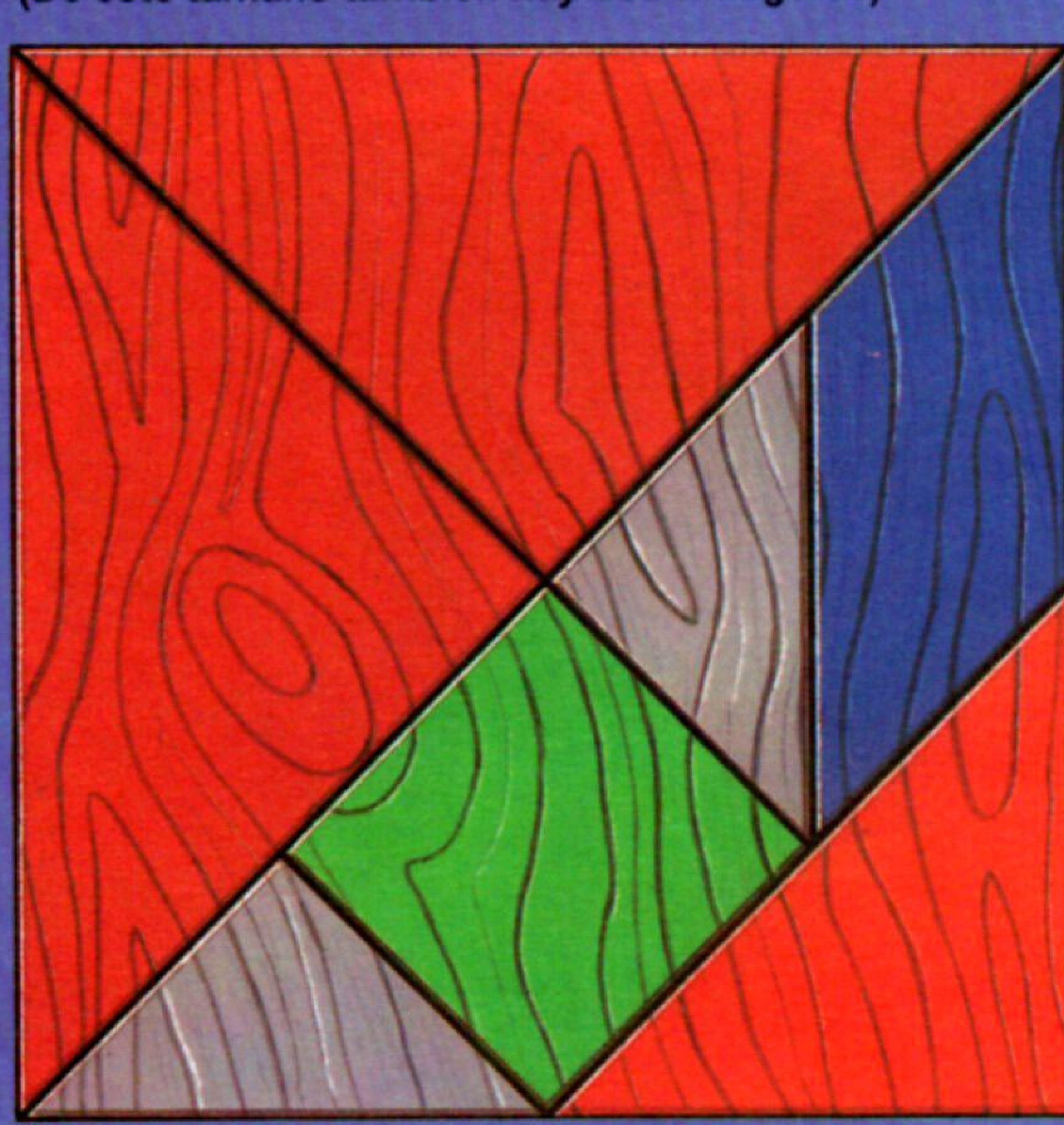
Primero se escribió el procedimiento PERRO, aun cuando todavía no existía ninguno de los procedi-

Dando forma

El puzzle tangram es un conjunto de siete formas que se pueden disponer en diversas combinaciones para crear diseños sencillos. Aquí listamos los procedimientos LOGO para dibujar las siete piezas básicas del tangram, así como un programa de demostración que dibuja la figura de un perro. El procedimiento PERRO empieza dibujando con el triángulo para la pata trasera del animal

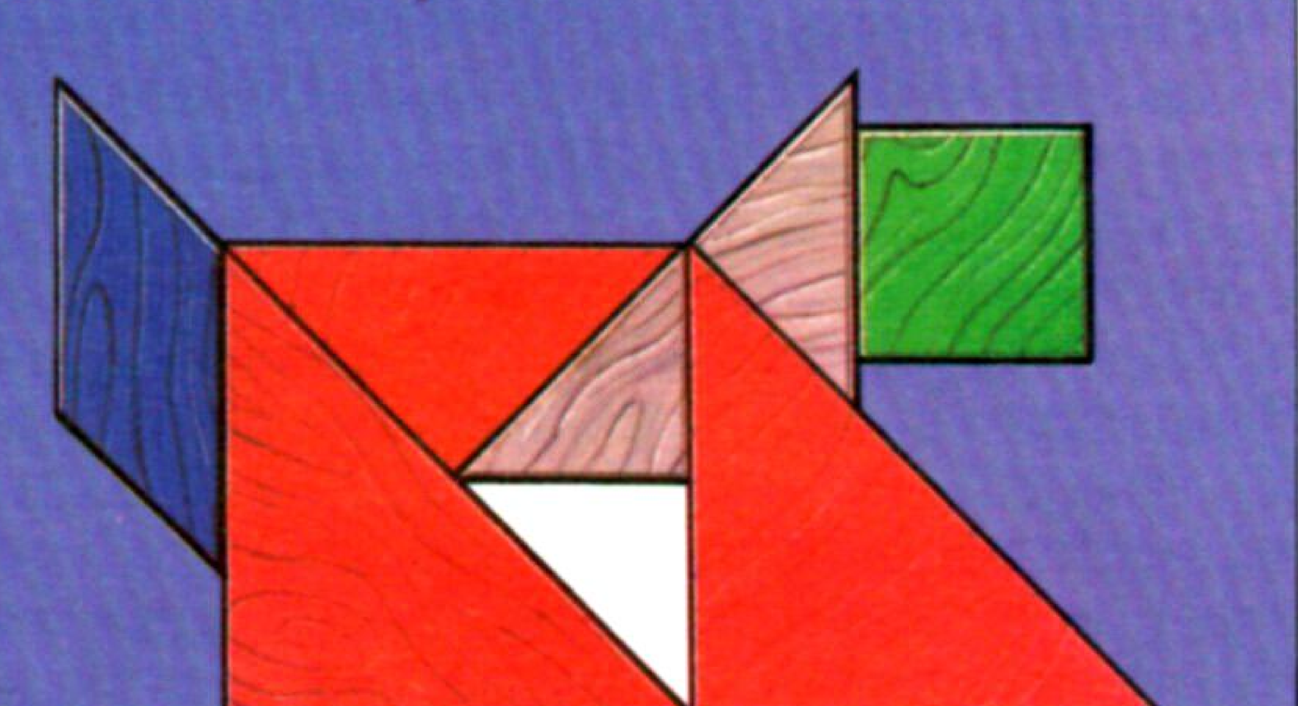
Procedimientos tangram

```
TO CUADRADO
  REPEAT 4 [FD 25 RT 90]
END
TO PAR
  REPEAT 2 [FD 25 RT 45 FD 35 RT 135]
END
TO TRI1
  FD 25 RT 135 FD 35 RT 135 FD 25 RT 90
END
(Hay dos triángulos de este tamaño)
TO TRI2
  FD 35 RT 135 FD 50 RT 135 FD 35 RT 90
END
TO TRI3
  FD 50 RT 135 FD 71 RT 135 FD 50 RT 90
END
(De este tamaño también hay dos triángulos)
```



Programa Perro

```
TO PERRO
  TRI3 MOVER1 PAR MOVER2 TRI2 MOVER3
  TRI1 MOVER4 TRI3 MOVER5 TRI1 MOVER6
  CUADRADO MOVER7
END
TO MOVER1
  PU FD 15 LT 45 PD
END
TO MOVER2
  PU RT 45 FD 35 LT 45 BK 35 PD
END
TO MOVER3
  PU LT 45 BK 25 PD
END
TO MOVER4
  PU RT 90 BK 25 PD
END
TO MOVER5
  PU FD 50 RT 45 PD
END
TO MOVER6
  PU FD 25 RT 135 FD 5 LT 90 PD
END
TO MOVER7
  PU LT 90 FD 5 RT 45 BK 25 RT 45
  BK 50 LT 90 BK 50 PD
END
```



Kevin Jones



mientos que lo componían. Después escribimos por separado cada uno de los procedimientos de dibujo de formas. A los mismos les siguieron los procedimientos de posicionamiento. Cada vez que se escribía un nuevo procedimiento, se ejecutaba PERRO para comprobar que todo se fuera acoplando correctamente. Cuando el LOGO llegaba a un procedimiento MOVER que aún no se hubiera escrito, se interrumpía con un mensaje de error. No obstante, era fácil decir a partir del dibujo si hasta ese momento todo era correcto o si existía algún error en el último procedimiento MOVER.

Nuestro conjunto de procedimientos demuestra otro punto importante: cada uno de los procedimientos de forma, y el propio procedimiento PERRO, deja el estado de la tortuga sin ninguna alteración. Es decir, al final del procedimiento la tortuga está en la misma posición y con el mismo encabezamiento que antes de que se ejecutara el procedimiento. Se dice que tales procedimientos son *transparentes*. Hacer que los procedimientos sean transparentes facilita la tarea de juntar procedimientos para construir dibujos más complejos. Tomemos el procedimiento PERRO, por ejemplo: una vez posicionada la tortuga sabemos que después de dibujar una pieza retornará a la posición en la que estaba cuando la empezó. De modo que no es necesario que sepamos nada acerca del funcionamiento interno de los procedimientos al objeto de colocar las piezas juntas. Haciendo que PERRO sea transparente conseguimos que sea más fácil utilizar este procedimiento como parte de otro; por ejemplo, podríamos dibujar una pantalla llena de perros.

Espacio de trabajo del LOGO

Ahora ya tendrá en la memoria del ordenador cierta cantidad de procedimientos, de modo que analicemos en mayor profundidad la organización de la memoria del LOGO. La memoria de trabajo se compone de una lista de *nodos* (cada uno de ellos de cinco bytes). Una vez cargado el LOGO, se dispone de 1 000 a 3 000 de éstos, según la máquina que emplee. A medida que se van definiendo procedimientos, estos nodos se consumen. También se utilizan más nodos cuando se ejecutan procedimientos o cuando se utilizan variables (que analizaremos más adelante en el curso).

Los procedimientos que se han definido constituyen el *espacio de trabajo*. Se puede ver qué procedimientos están retenidos en el espacio de trabajo entrando el comando POTS (por PRINTOUT TITLES: imprimir títulos). Para ver un procedimiento en particular, emplee PO (por PRINTOUT: imprimir); por ejemplo, PO CUADRADO. Si un procedimiento ya no se necesita más, se puede liberar el espacio de trabajo que ocupa mediante la utilización de ERASE: la instrucción o comando ERASE CUADRADO eliminaría de la memoria el procedimiento llamado CUADRADO. Borrar un procedimiento libera los nodos que éste utiliza. El LOGO marcará estos nodos pero no los agregará todavía a la lista de nodos libres; en cambio, seguirá trabajando con su lista actual de nodos libres hasta que se consuman todos éstos. Entonces repasará su memoria, reuniendo todos los nodos que se hayan liberado y los utilizará para formar una nueva lista de nodos libres. A este proceso se alude como *recolección de basura (garbage collection)* y es el motivo por el cual el LOGO parece dudar uno o dos segundos de vez en cuando.

Cómo salvar procedimientos

Con el fin de almacenar permanentemente los procedimientos en disco, usted debe salvar su espacio de trabajo como un archivo. Utilizando MISPROCS como ejemplo de nombre de un archivo, debería entrarse el comando SAVE "MISPROCS (observe las comillas: solamente abren, no cierran). El espacio de trabajo propiamente dicho no sufrirá ninguna alteración por esto. El archivo se podría cargar con READ "MISPROCS". Esto hace que se definan los procedimientos que contiene el archivo y se agreguen al espacio de trabajo en curso. Si se define un procedimiento con el mismo nombre que otro ya retenido en el espacio de trabajo, entonces la nueva definición reemplaza a la anterior.

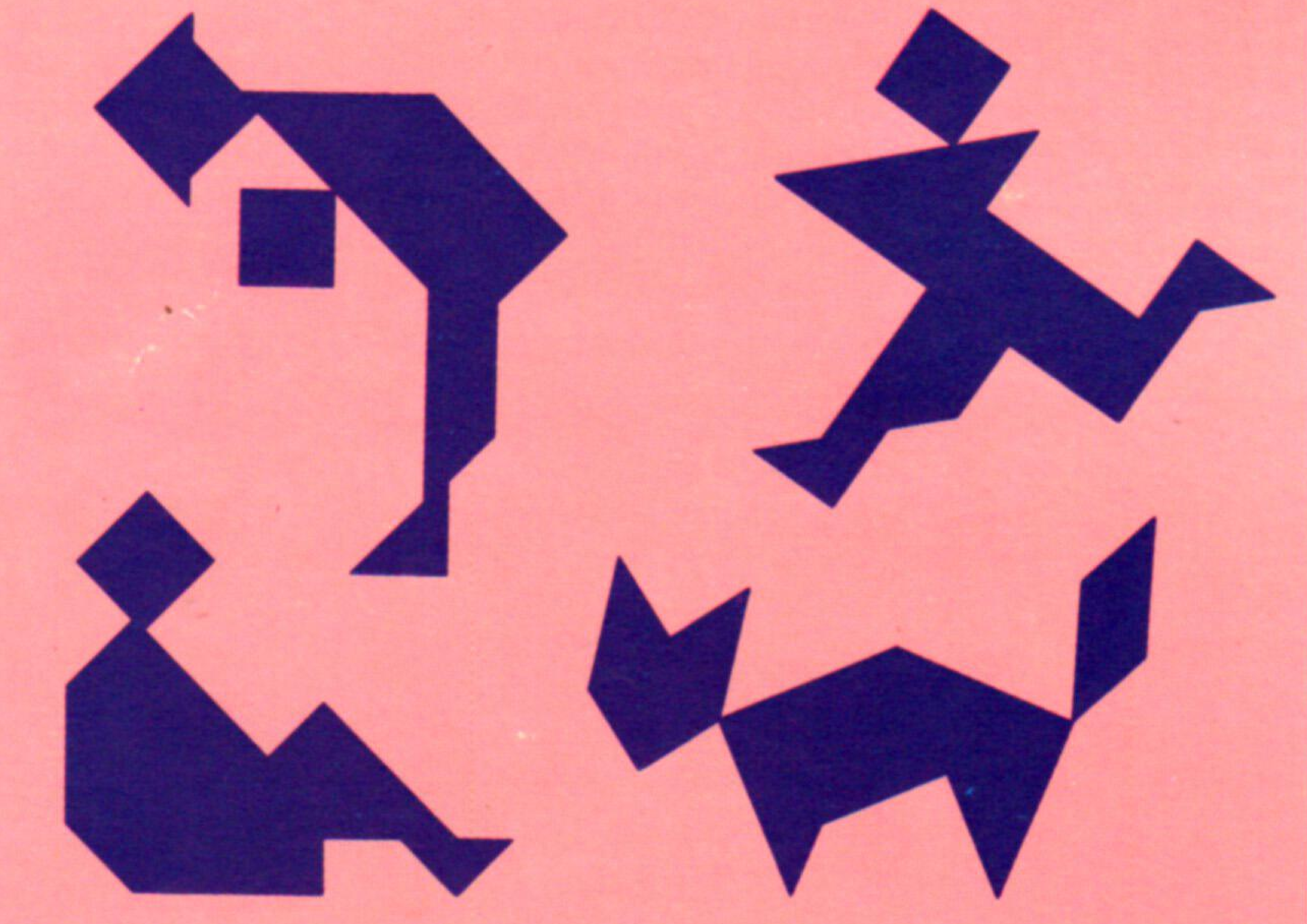
Otras instrucciones o comandos útiles para la manipulación de disco son CATALOG y ERASEFILE. CATALOG (catálogo) da una lista de todos los archivos del disco, y ERASEFILE "MISPROCS" borraría el archivo MISPROCS del disco. Las versiones de LOGO basadas en cassette usan instrucciones diferentes.

Abreviaturas

ERASE	ER
PRINTOUT	PO
PRINTOUT TITLES	POTS

Problemas con procedimientos

1) Escribir procedimientos para las otras formas tangram ilustradas. (Primero tendrá que resolver el puzzle de cómo construir la forma a partir de las distintas piezas, ¡por supuesto!) **2)** Escribir un procedimiento para dibujar una "casa" (un triángulo equilátero encima de un cuadrado) **3)** Escribir un procedimiento para dibujar un tablero de cuadrados de cinco por cinco **4)** Reescribir el procedimiento empleado anteriormente para dibujar una estrella de seis puntas de modo que utilice subprocedimientos



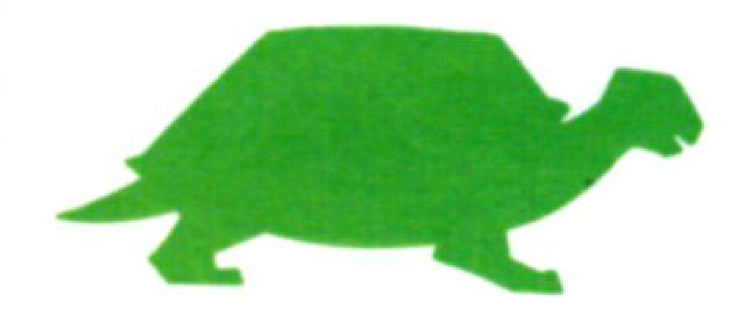
Liz Dixon

Complementos al LOGO

En todas las versiones LCSI, los nombres de los procedimientos deben ir precedidos por comillas si son argumentos de PO o ERASE: por ejemplo, PO "CUADRADO y ERASE "CUADRADO.

Para leer un archivo en disco use LOAD "MISPROCS.

Las versiones basadas en cintas de cassette o microdrives poseen instrucciones algo diferentes a aquellas basadas en disco. Consulte el manual.



Biblioteca pública

La creación de bibliotecas o librerías de rutinas de uso general permite optimizar los esfuerzos de programación

Seguir los métodos de diseño estructurado que ya hemos descrito en este curso puede parecer un enfoque algo largo; pero, en realidad, ahorra tiempo (no sólo en la codificación sino especialmente en la depuración de un programa). Ello se debe a que los programas que se crean directamente desde el teclado tienden a tener estructuras y algoritmos innecesariamente complicados. Esto significa que se tarda más tiempo en escribirlos, son más susceptibles a la aparición de errores y, dado que son más difíciles de seguir, requieren mucho más esfuerzo para su comprobación y depuración. Planificar el programa por adelantado simplifica la estructura y los algoritmos y, por consiguiente, conduce a menos errores de codificación y una comprobación y depuración más sencillas.

Aún más importante, diseñar por anticipado le ahorra al programador el escribir una estructura de control o de archivo que posteriormente pueda resultar inadecuada (quizá por no haber destinado espacio suficiente para un campo del archivo). Los problemas como éste, que son fundamentales para la forma en que opera el programa, pueden determinar que se hayan de reescribir porciones muy importantes del mismo.

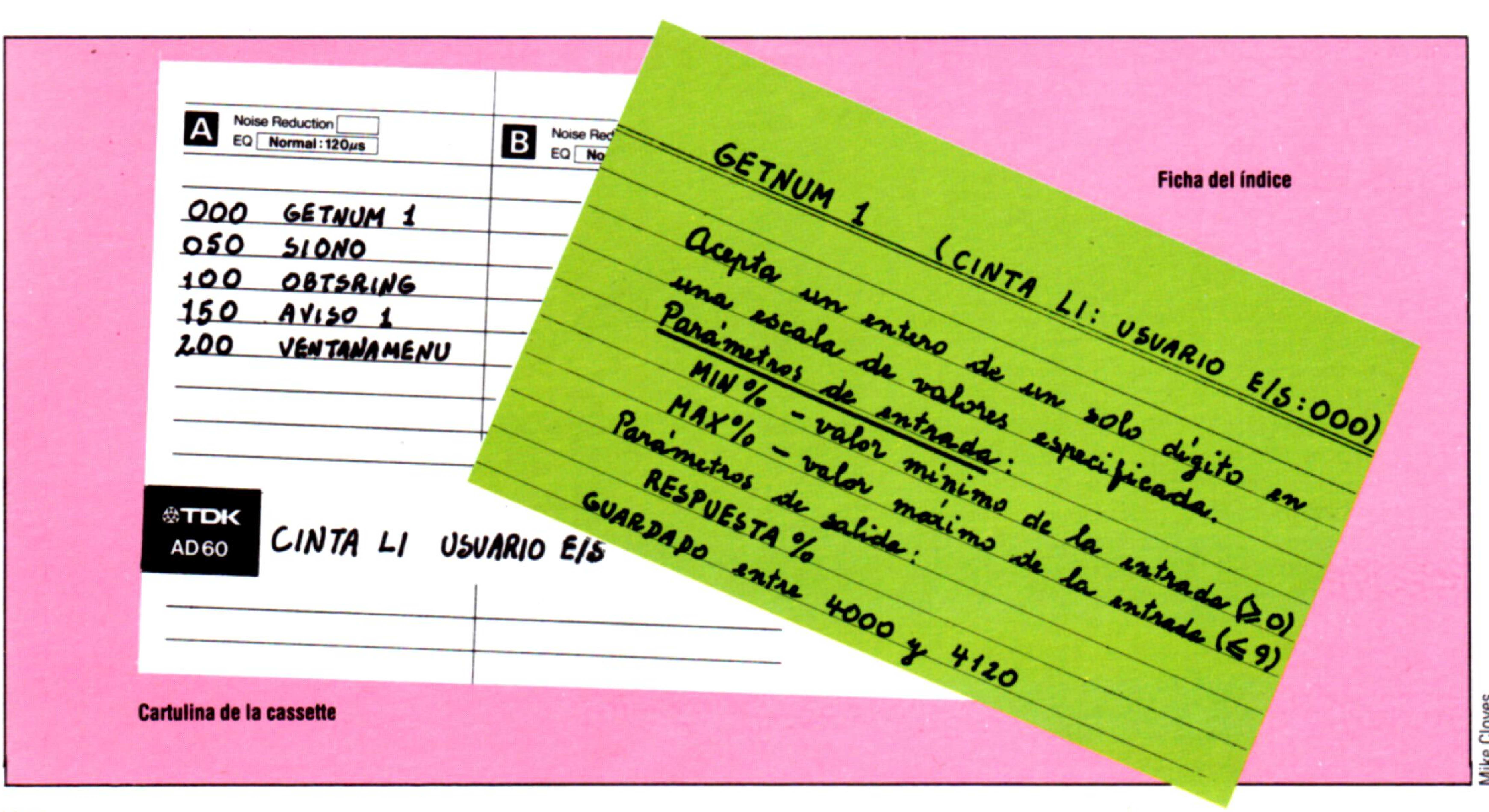
Sistema uniforme
Las bibliotecas de subrutinas no tienen ninguna utilidad si carecen de un sistema de documentación homogéneo. Esto es particularmente cierto en el caso de los usuarios de cassettes: inspeccionar el contenido de una cassette indocumentada cargando y listando cada programa es una tarea ímproba

Quienes posean un teclado "apropiado" tipo máquina de escribir tal vez deseen invertir algo de tiempo aprendiendo mecanografía al tacto. Aparte de esto, sin embargo, es poco lo que se puede hacer para aumentar la velocidad a la cual se escriben en el teclado las líneas del programa. No obstante, el proceso de codificar programas se puede acelerar de diversas maneras. La primera es la más sencilla: definir, adoptar y utilizar cierto número de "convenciones" cuando se codifica. Tales medidas incluyen: hacer uso de determinados tipos de nombres para variables locales, para diferenciarlos de las variables del programa principal o variables globales; empezar cada subrutina en líneas que acaben en 000; acabar cada subrutina con el RETURN en una línea propia; empezar cada tipo de subrutina en un bloque determinado de líneas (rutinas para tratamiento de archivos entre la 9000 y la 9999, utilidades, de la 50000 en adelante, etc.).

La utilización de estas convenciones reporta numerosos beneficios: uno no tiene que ir a la pesca de las rutinas del menú porque sabe que siempre están en el mismo sitio, ni tiene que preocuparse de si ya ha empleado o no el mismo nombre de variable en el programa principal y en una subrutina, porque su nombre indicará que se trata de una variable local.

Bibliotecas de programas

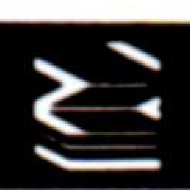
Tales técnicas de codificación son asimismo útiles cuando se crean bibliotecas de programas. Una biblioteca de subrutinas bien organizada puede significar un ahorro de tiempo de codificación para un programa largo de hasta el 50 %. La mejor forma de comenzar una biblioteca de este tipo es remitirse a los programas existentes y extraer todas las subru-



Ficha del índice

Cartulina de la cassette

Mike Cloves



tinas que estén bien escritas y posean alguna aplicabilidad general (rutinas de E/S, rutinas de verificación de datos, conversión de mayúsculas a minúsculas, etc.). Cada rutina se debe guardar como un archivo separado y éstos se deben agrupar entre sí de acuerdo a su función (si se van a almacenar en cinta entonces cada grupo de funciones se debe almacenar en una cassette separada), con nombres de archivo que tengan algún sentido para identificarlos. Lleve un índice de fichas o una base de datos de los nombres de los archivos, junto con una descripción de qué hace cada rutina.

Huelga decir lo importante que es asegurarse de que todas las rutinas de la biblioteca estén rigurosamente comprobadas y depuradas. Las mismas se habrán de utilizar en programas para los cuales no fueron diseñadas especialmente, de modo que asegúrese de que detectarán cualquier valor de entrada ilegal. Asimismo, debe asegurarse de que ninguna salida de valores de las rutinas de la biblioteca suponga problemas para el programa que las emplea. Haga a cada rutina tan eficaz como sea posible e incluya toda la documentación interna que sea necesaria para que se entienda la función de la rutina cuando sea utilizada en el futuro. Agregue una rutina al grupo cuando surja la necesidad: no tiene ningún sentido agregar nuevas rutinas demasiado específicas porque la experiencia demuestra que esto es un esfuerzo en gran medida desperdiciado. No se olvide de numerar las líneas de las rutinas de la biblioteca según la convención establecida (ello le evitará la RENUMeración cuando las rutinas se fusionen en un nuevo programa). En las revistas de informática, que con frecuencia publican listados de rutinas así como programas completos, se pueden encontrar eficaces rutinas de biblioteca.

Para hacer uso de una biblioteca como ésta es necesario disponer de una forma de fusionar rutinas entre sí para formar un programa completo. A quienes utilizan lenguajes compilados se les suele proporcionar un programa "montado" *(link-editor)* o similar; éste toma módulos compilados y los une para hacer un programa ejecutable. Para los programadores de BASIC, a menos que dispongan de un compilador, la forma más sencilla de conseguir esto consiste en emplear una combinación de instrucciones RENUM (renumerar) y MERGE (fusionar). Para fusionar una rutina de biblioteca en un nuevo programa, primero cargue el programa, decida dónde irá la rutina de biblioteca y asegúrese de que haya un bloque de números de línea sin utilizar suficiente como para darle cabida. De ser necesario, RENUMere la rutina de biblioteca de modo que encaje en el espacio que tiene asignado. Luego emplee la instrucción MERGE para juntar los dos programas; compruebe que todo funcione correctamente y guarde (SAVE) el nuevo programa con la rutina de biblioteca en su sitio.

Esfuerzos en grupo

Con frecuencia se da el caso de que los usuarios de ordenadores personales trabajan en grupo para escribir programas, ya sea en la escuela o en sus clubes de usuarios. La mayor parte de cuanto se ha dicho acerca del diseño de programas y de la eficacia del programador cobra especial relevancia para tales esfuerzos de equipo. En realidad, la mayoría de estas ideas y el concepto de programación estructurada se desarrollaron con el fin de dividir el peso del trabajo en los proyectos de programación comerciales. Por consiguiente, distintos programadores podrían trabajar en diferentes partes del mismo programa al mismo tiempo para producir un programa operativo. Para que los programadores de BASIC trabajen de este modo, es esencial estar de acuerdo acerca de las convenciones a utilizar para la codificación. Suponiendo que se ha llegado a un acuerdo en cuanto al diseño, el programador de un módulo individual necesita saber:

1) La estructura y la organización de los archivos.
2) Las convenciones que se han acordado para la nomenclatura de las variables. A las variables más importantes, como, por ejemplo, las matrices que se utilicen a lo largo del programa, se les debe dar un nombre por anticipado. Se deberá acordar una convención para los nombres de las variables locales. A las variables que se pasen de un módulo a otro se les debe dar un nombre por anticipado o bien se debe diseñar una forma de asegurar que cada una sea única (agregándole el número del módulo original como sufijo, por ejemplo).
3) Cuáles son las rutinas de biblioteca que están a disposición del grupo, cuál es el formato de cada una de ellas, cómo se designan sus variables, qué hacen y en qué medida están comprobadas y depuradas.
4) Organización de las rutinas de tratamiento de errores (por ejemplo, si cada rutina se ocupa de sus propios errores o si las rutinas establecen un indicador de error, que es entonces tratado por la rutina de control).
5) La función exacta de cada módulo que se está escribiendo.
6) La gama y el tipo exactos de datos que cada módulo individual aceptará como entrada y devolverá como salida.

Esto supone un largo período de planificación con muchos encuentros para acordar la estrategia, seguido de un corto período de programación. De la comprobación nos ocuparemos más adelante.

MERGE

- El Spectrum posee la versión inmediata de la instrucción: fusiona el archivo especificado con el programa en memoria; en caso de coincidencia de números de línea, la línea entrante se escribe encima de la ya existente.
- Con la instrucción *SPOOL del BBC Micro se pueden crear versiones ASCII de los archivos de programas, luego escribir un programa en BASIC (o utilizar un procesador de textos) para acceder a estos archivos, a una línea de programa cada vez. Fusione los dos archivos en un tercer archivo ASCII y conviértalo en un programa utilizando la instrucción *EXEC.
- En el Commodore: OPEN 1,1:CMD1:LIST:PRINT #1:CLOSE 1 crea en cinta un archivo ASCII sin nombre del programa que esté cargado en memoria. Cargue (LOAD) el otro programa y añádale una rutina para entrar (INPUT) e imprimir las líneas del archivo ASCII en pantalla

Blanca elegancia

Esta vez examinamos el nuevo ordenador Apple II*c* y analizamos la estrategia de marketing de la empresa

El éxito del Macintosh y la creciente competencia (por parte de Commodore en el mercado norteamericano del ordenador personal y de empresas como ACT e IBM en el mercado internacional de gestión) ha creado algunas dudas acerca del futuro de la gama de ordenadores Apple. Muchos distribuidores y analistas industriales vaticinaron que la gama se estaba aproximando al final de su vida en el mercado, a pesar de la insistencia de Apple en el sentido de que permanecería comprometida con la máquina 6502 y su gran base de usuarios. Para demostrar su apoyo, la empresa ha lanzado recientemente el Apple II*c*, así como mejoras de software y hardware para las más antiguas líneas Apple II. Se espera que los nuevos productos prolonguen la vida del Apple II en el mercado en unos tres años.

En sus diversas formas (II, II+ y II*e*), el Apple II ha ayudado a crear el mercado del ordenador personal, ha dominado las ventas de ordenadores en Estados Unidos durante varios años y ha contribuido a que las ventas totales de Apple superaran el billón de dólares. Existen más de dos millones de ordenadores Apple en uso en todo el mundo, a pesar de lo cual el Apple II jamás ha alcanzado el mismo nivel de éxito de ventas en el ámbito europeo, básicamente debido a una ineficaz política de precios y de comercialización. El precio de venta al público de la máquina era excesivamente elevado como para que se la considerara un ordenador personal. Y con frecuencia se alude a interferencias de las oficinas centrales de Apple en California como la razón por la cual Apple nunca ha conseguido el tipo de participación en los mercados británicos de gestión o educativo que ha obtenido en Estados Unidos. No obstante, el relativamente pequeño grupo de usuarios Apple de Gran Bretaña tiende a ser sumamente leal a la máquina.

Mejora portátil
La nueva versión del Apple II, mejorada y portátil, es el II*c*. Tiene 128 K de RAM, una visualización de 80 columnas, una variedad de interfaces y una unidad de disco incorporada. En la fotografía lo vemos con la pantalla opcional de fósforo verde



Chris Stevens

La última reencarnación del Apple II es el II*c* (la *c* alude a “compacto”). Es aproximadamente la mitad en tamaño que sus antecesores, a pesar de lo cual alberga una unidad de disco de 5 1/4 pulgadas a media altura a uno de los lados de su carcasa. Con su peso de 3,4 kg, el II*c* es transportable y está claramente diseñado para utilizarlo durante el día en el trabajo y llevarlo luego a casa por la noche. Con este fin, el II*c* tiene una pequeña asa para transporte moldeada en su carcasa plástica y una serie de conectores (para utilizar con una pantalla compuesta o RGB en el trabajo y un televisor normal en casa). El asa se acomoda en la carcasa para conformar un ángulo de trabajo cómodo. También hace que el aire circule alrededor de la máquina para evitar el sobrecalentamiento.

A diferencia de los modelos Apple II anteriores, el II*c* es un sistema cerrado, sin ranuras para ampliación en su interior. En cambio, Apple ha incorporado en la máquina varias opciones de las más importantes. Éstas incluyen las puertas para visualización en pantalla y televisor; una puerta para palanca de mando que también soporta el ratón opcionalmente; una puerta para modem; una puerta para impresora; un conector para salida de audio, y un conector para una segunda unidad de disco. Las interfaces están etiquetadas con iconos (pequeñas representaciones gráficas de su función). El II*c* también posee incorporada una visualización de 80 columnas y 128 Kbytes de RAM. La mayoría de estas características son opcionales en el II*e* y exigirían la adición de al menos tres placas de ampliación.

El Apple II*c* posee un teclado QWERTY de 63 teclas, con un trazado similar al del Apple II*e*. La tecla Reset, sin embargo, se ha desplazado a una posición por encima del borde izquierdo del teclado y junto a ella se han colocado dos pequeños interruptores. El interruptor izquierdo conmuta la visualización en pantalla de 40 a 80 columnas. El manual del usuario recomienda utilizar una visualización de 40 columnas cuando se trabaje con un televisor, y una visualización de 80 columnas para una pantalla. (Parte del software Apple existente visualizará sólo 40, independientemente de la posición del interruptor.) El segundo interruptor conmuta

entre el juego de caracteres europeo y los caracteres norteamericanos que aparecen en el teclado. Esto es útil cuando se requiere un carácter que sólo se puede encontrar en uno de los dos juegos (como "#", que en el teclado europeo está reemplazado por "£"). Encima del borde derecho del teclado hay dos luces: una indica que el equipo está conectado a la red y la otra se ilumina cuando se está utilizando la unidad de disco.

Cuando se conecta el II*c*, la unidad de disco comienza a girar automáticamente y espera un disco. Continuará girando hasta que se cargue un disco o hasta que se pulse la tecla Reset manteniendo deprimida la tecla Control. Sin ningún disco en la unidad, la II*c* carga el BASIC Applesoft desde la ROM. El Applesoft permanece virtualmente sin modificaciones desde que se introdujera el Apple II+. Sólo tiene unas ligeras variaciones respecto a las primeras versiones del BASIC Microsoft estándar. La utilización del Applesoft permite que el II*c* ejecute programas escritos para los modelos anteriores. Lamentablemente, el Applesoft carece de muchas de las estructuras de programación de que disponen las versiones más avanzadas, como el BASIC BBC. Por ejemplo, el Applesoft no posee la instrucción RANDOMIZE, facilidad para autonumeración de líneas, estructura IF...THEN...ELSE ni instrucción WHILE. También carece de las instrucciones CIRCLE y PAINT para la programación de gráficos.

Cuando en la unidad de disco hay un disco, el II*c* ejecuta el DOS 3.3 (discos Apple II+y II*e*) o bien el PRODOS, el nuevo sistema operativo de Apple. Éste es un derivado del sistema operativo que Apple diseñó para su primer sistema de gestión, el Apple III. El PRODOS posee una estructura de archivos jerárquica (árbol). Los archivos en disco se almacenan de forma muy parecida a los documentos que se guardan en un mueble archivador. Por consiguiente, los archivos relativos, por ejemplo, al proyecto ZED se podrían archivar en el disco bajo el encabezamiento ZED. Los archivos de contabilidad del proyecto ZED (como Costos, Ventas, Ganancias) se podrían reunir en un grupo denominado Cuentas. Con un sistema de archivo manual de este tipo, cuando se deseara hallar el archivo para Ventas, debería primero abrirse el archivo principal, ZED, después abrir el archivo Cuentas, y finalmente extraer el archivo etiquetado como Ventas. En el PRODOS esto se realiza a través de un "nombre de encaminamiento" *(pathname)* que lista los nombres de archivo apropiados por orden, de modo que el proceso que hemos descrito se cubriría entrando los nombres de los archivos, separados mediante barras:

/ZED/CUENTAS/VENTAS/

Los nombres de encaminamiento pueden ser de hasta 64 caracteres de longitud. Este proceso puede parecer complicado, y de hecho lleva un tiempo acostumbrarse a él, pero a la larga simplifica la organización y administración de los archivos en disco. Los sistemas de archivo de árbol como el PRODOS también se utilizan en el MS/DOS, el sistema operativo que emplea el IBM PC.

La visualización en pantalla del II*c* también representa una mejora en relación a los modelos anteriores. Aparte de dos opciones de texto (24 líneas por 40 u 80 caracteres), el II*c* posee tres pantallas para gráficos: 40 por 40 (baja resolución), 280 por

Un relato ejemplar



Chris Stevens

Apple III

El Lisa

Comprendiendo la importancia del mercado del ordenador de oficina, Apple desarrolló el Apple III, una brillante máquina de mesa que hace uso de dos procesadores 6502 para soportar hasta 512 K de memoria. El Apple III dispone de un sistema operativo llamado SOS (*Sophisticated Operating System:* sistema operativo sofisticado), que le permite comunicarse con una unidad de disco rígido y almacenar archivos en una estructura jerárquica. Esta estructura de árbol constituyó la base conceptual para el sistema operativo del Lisa, y muchas de sus características aparecieron en el MS/DOS del IBM PC. Lamentablemente, el III tuvo algunas dificultades de hardware inmediatamente después de su lanzamiento y eso le creó una mala reputación. Apple renovó y sustituyó todas las máquinas defectuosas, pero el III jamás superó la mala publicidad de que fue objeto ni la sensación general de que el sistema operativo era demasiado complicado. Como resultado, el Apple III desapareció del mercado.

El Lisa, introducido en 1983, fue el primer ordenador personal para el mercado masivo que utilizó software integrado con ventanas, tenía un sistema operativo basado en imágenes en vez de en palabras y se operaba mediante un dispositivo de control manual: el ratón. El Lisa suscitó tal expectación que hizo que la cotización en bolsa de la empresa casi se triplicara en unos pocos meses. Desarrollado a un costo de más de 50 millones de dólares, se le fijó un precio demasiado elevado y fue dirigido al mercado equivocado. Originalmente Apple dirigió el Lisa a los ejecutivos de las grandes corporaciones. Mientras algunos altos ejecutivos se han sentido muy impresionados por el Lisa, la mayoría de sus ventas se han ido a pequeñas compañías especializadas en publicidad, diseño gráfico y relaciones públicas. El Lisa no se vendió tan bien como se esperaba, y la cotización de las acciones de Apple descendió más allá de su valor anterior al lanzamiento del ordenador nuevamente en el lapso de unos pocos meses. Recientemente el Lisa ha sido reemplazado por el menos caro Lisa II. Bajo la guía de John Scully, venido de la multinacional Pepsi Cola para conducir Apple, la innovadora empresa parece haber aprendido de sus errores. El Macintosh se está vendiendo en enormes cantidades en todo el mundo, recuperando algunos de los costos de desarrollo del Lisa/Mac, y el II*c* parece llamado a darle renovados bríos al principal producto de Apple, el Apple II.

192 (alta resolución) y 560 por 192, llamada Double Hi-Res (alta resolución doble). Hay 16 colores disponibles. Apple ofrece una pantalla de fósforo verde y se espera que introduzca pronto una visualización en cristal líquido (LCD). La pantalla LCD la fabricará Sharp para Apple y tendrá una visualización completa de 24 líneas por 80 caracteres. Se espera que pronto habrá disponible un paquete de pilas para el sistema, que permitirá que el Apple II*c* se convierta en un ordenador totalmente portátil.

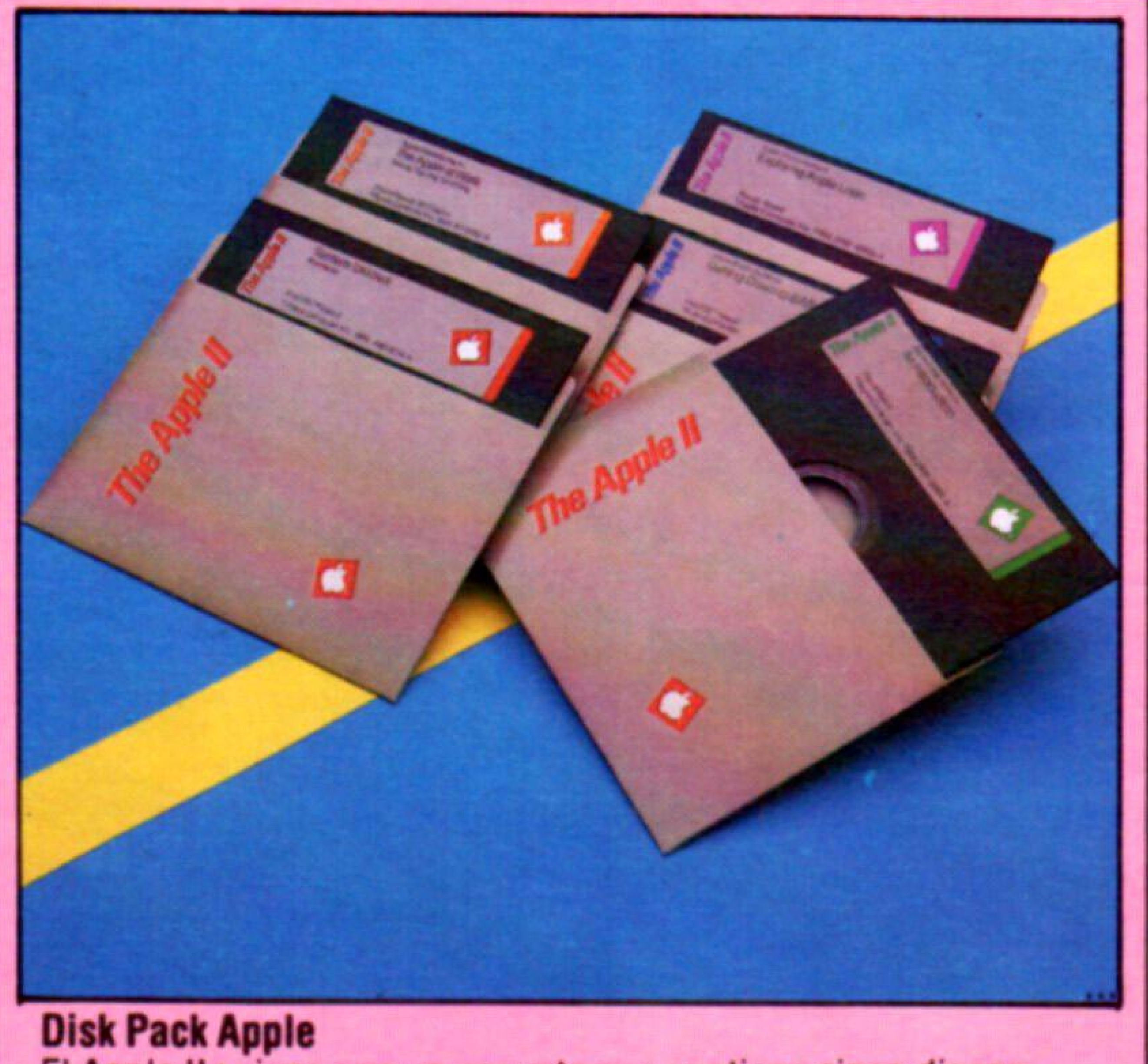
La mayor cualidad que tiene el II*c* es su base de software. Para el Apple II se han escrito más de 17 000 programas. Si bien algunos de éstos sólo se venden en Norteamérica, uno todavía puede estar bastante seguro de que cualquier cosa que desee hacer con un Apple seguramente ya se habrá hecho y se habrá escrito el software adecuado. La base de software incluye algunos de los mejores programas de juegos del mundo (CHESS 7.0, ZORK, Simulador de vuelo Microsoft, Pinball Construction Set); una amplia variedad de programas de tratamiento de textos, hoja electrónica y base de datos; programas de contabilidad; programas para diseño de gráficos; programas científicos para control de laboratorio y programas educativos (desde libros de lectura para principiantes hasta cálculo avanzado).

Además del software II y II*e* existente, Apple ha introducido un programa llamado Appleworks, un programa integrado de tratamiento de textos, hoja electrónica y base de datos con ventanas. Appleworks es bastante sofisticado y fácil de utilizar. El II*c* viene con un disco de introducción al Appleworks, si bien no se trata de un ejemplar operativo del programa. Como es típico tratándose de Apple, la empresa da por sentado que finalmente usted deseará adquirir este paquete. Otros discos que se suministran con el sistema son: una introducción similar al LOGO Apple; Apple Presents Apple, introducción interactiva al sistema básico; una introducción muy simple a la programación en BASIC, y el disco de utilidades de sistemas PRODOS. También está disponible el MousePaint, un programa de dibujo activado por ratón basado en el MacPaint. El MousePaint se suministra con el ratón Apple II.

El II*c* viene con un pequeño folleto que describe cómo instalar el sistema y una guía del usuario de 142 páginas que explica breve y claramente las operaciones del sistema y el empleo de los cinco discos que vienen con el ordenador. Los manuales están bien escritos e ilustrados a todo color. Es evidente que están pensados para el usuario novel.

El disco del II*c* es muy atractivo y elegante. Apple abandonó el plástico beige utilizado para el II, el Macintosh y el Lisa en favor de un acabado blanco brillante. Las estrías que atraviesan la parte superior de la carcasa dejan que el aire fluya por los circuitos para mantener frío el sistema.

El Apple II*c*, al igual que sus predecesores, es una gran máquina de sobremesa para la oficina. Con el agregado de la pantalla LCD y el paquete de pilas, cuya aparición se espera en breve, habrá de ganarse una reputación como compañero de trabajo útil y portátil. Si su precio hubiera sido más asequible, también podría haber sido un popular ordenador personal.



Disk Pack Apple
El Apple II*c* viene con un paquete que contiene cinco discos. Cuatro de ellos son una introducción al funcionamiento de la máquina, la programación en BASIC y a programas de aplicaciones opcionales que Apple espera que uno compre. El quinto es el disco de operaciones del sistema con el PRODOS, el nuevo sistema de archivos en disco para la línea Apple II

Salida de video compuesto

Puerta para impresora RS232

Fuente de alimentación
Es de 12 voltios pero aun así requiere un transformador para entrada de corriente de 240 voltios

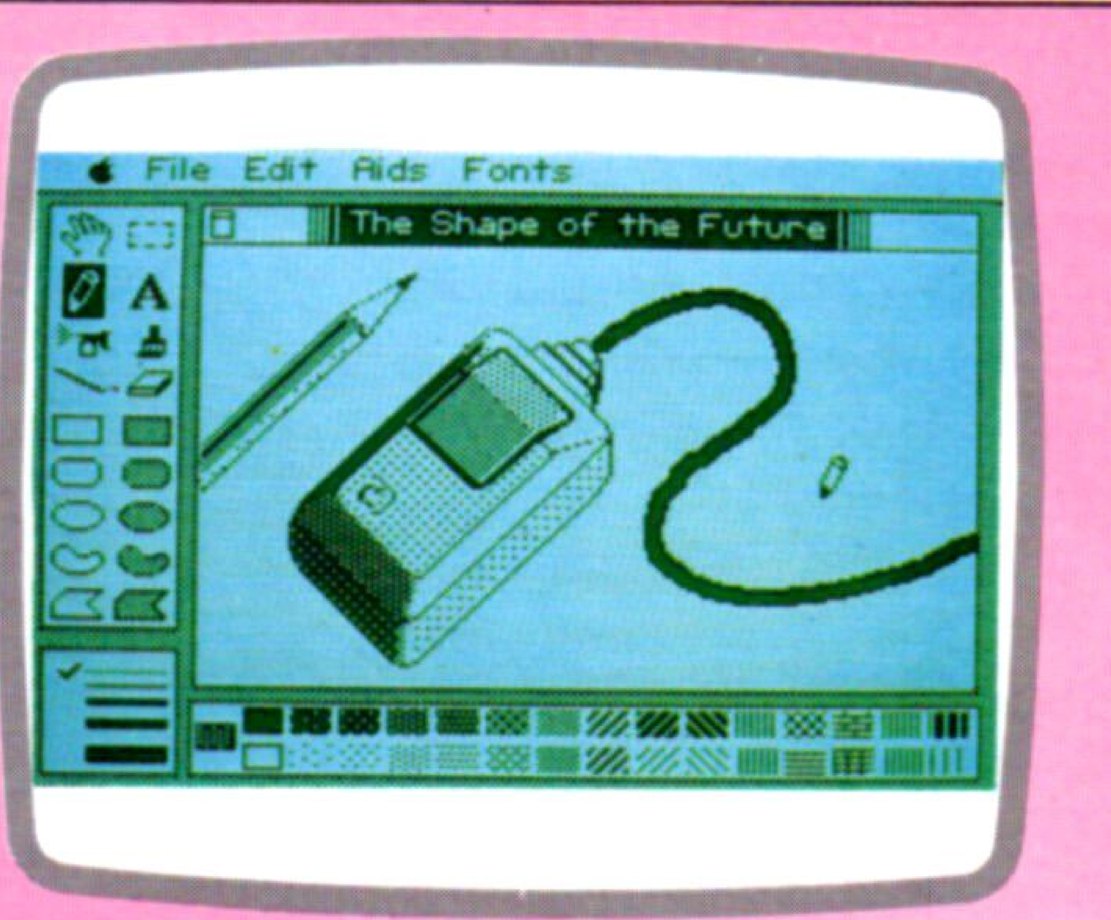
Salida video RGB
Enchufando un pequeño adaptador PAL en esta puerta, se puede conectar el II*c* a un aparato de televisión estándar

Controladores entrada-salida
Estos chips controlan las operaciones del teclado y las puertas de entrada y salida

Conector audio
Permite conectar al II*c* con un amplificador hi-fi externo

ROM
Retiene el BASIC Applesoft y las rutinas del sistema





MousePaint
El ratón Apple II está disponible para los modelos II+, II*e* y II*c* y viene con MousePaint. MousePaint está basado en el MacPaint, pero se trata de una versión a escala reducida del programa para el Macintosh. Permite que se utilice el ratón para crear imágenes muy fácilmente, y está escrito para sacar partido de la visualización en pantalla de alta resolución del II*c*. Para utilizar el ratón con otros ordenadores Apple II, hay que adquirir una placa de interface extra que se enchufa en una de las ranuras para ampliación internas



Unidad de disco
La unidad de disco incorporada de 143 K es compatible con la mayoría de los discos Apple II+ y II*e*

Puerta para unidad externa
Aquí se puede conectar una segunda unidad de disco

Puerta RS232 para modem

Control manual
Esta puerta admite una palanca de mando o el ratón opcional

128 K de RAM para el usuario
Esta cantidad es el doble de la que viene como estándar con el Apple II*e*

CPU
El chip 65C02 es una versión CMOS del 6502. Requiere menos potencia que otras versiones del chip, de modo que puede operar con una pila

APPLE II

DIMENSIONES
305×292×64 mm

CPU
6502 @ 1MHz

MEMORIA
128 K de RAM, 16 K de ROM

PANTALLA
24 líneas de 40 u 80 caracteres. Tres modalidades para visualización de gráficos con resolución máxima de 560×192 pixels y 16 colores

INTERFACES
Puerta de 9 patillas para palanca de mando que también funciona como puerta para ratón; puerta RS232 para modem; salida RGB (o PAL TV); salida para video compuesto; puerta para unidad de disco exterior, y puerta RS232 para impresora

LENGUAJES DISPONIBLES
BASIC Applesoft residente en ROM; LOGO, PASCAL, FORTRAN

TECLADO
Teclado de 63 teclas tipo máquina de escribir con cuatro teclas para cursor, juegos de caracteres internacionales

DOCUMENTACION
Con la máquina viene una guía del usuario a todo color y fácil de comprender. La guía está claramente diseñada para el usuario novel. Asimismo, hay dos delgados manuales para ayudar al usuario a instalar el II*c* y trabajar con el disco de utilidades del sistema

VENTAJAS
La mayor cualidad del II*c* la constituye su compatibilidad con el Apple II, lo que significa que puede ejecutar los más de 17 000 títulos disponibles para el otro Apple

DESVENTAJAS
El BASIC Applesoft, que en seis años no ha sufrido ninguna modificación sustancial y está empezando a hacer ostensible su edad, carece de flexibilidad. Puede que el precio del II*c* lo coloque fuera del alcance de muchos usuarios personales



El mejor reparto

He aquí un programa que le permite crear su propio juego de caracteres para el Commodore 64

El Commodore 64 es capaz de producir espléndidos gráficos y sonido (tal como lo demuestra ampliamente gran parte del software comercial), pero su BASIC no dispone de ninguna instrucción de color ni de sonido incorporada con ese fin. Las instrucciones BEEP, DRAW, INK y PAPER de que dispone el BASIC del Spectrum, por ejemplo, carecen de equivalentes en el magro juego de instrucciones que tiene a su disposición el programador del Commodore 64. El resultado es que la mayoría de los programas en BASIC poseen sonido y gráficos muy elementales e incluso los mejores programas tienden a contener muchas sentencias DATA y POKE, tal como refleja el listado incluido en este capítulo. El programa generador de caracteres que listamos aquí hace que el proceso de definir nuevos caracteres sea menos penoso al permitir que el usuario los diseñe en la pantalla, en vez de colocar (POKE) valores directamente en la RAM; estas definiciones hechas en pantalla se colocan (POKE) luego automáticamente en la memoria.

Ya hemos investigado con cierto detalle el procedimiento que implica definir sus propios caracteres en el Commodore 64 (véase p. 712). Las acciones preliminares esenciales se realizan en la subrutina de la línea 61000. El tope de la memoria para el usuario se baja desde la posición 40959 a la 14335. El juego de caracteres en mayúsculas completo de dos Kbytes que el Commodore tiene residente en ROM (desde la dirección 53248 en adelante) se copia luego en RAM (desde 14336 en adelante), donde se puede acceder y manipular utilizando sentencias PEEK y POKE. Por último, se conecta el VIC (Video Interface Chip) para direccionar el juego de caracteres reubicado.

Una vez reubicado el juego de caracteres en RAM, la rutina de inicialización visualiza en la pantalla dos "ventanas" y el control pasa a la rutina de entrada de la línea 2500. Esta rutina explora el teclado y mantiene un cursor intermitente en la ventana izquierda o "de edición". En esta ventana se visualiza el carácter que se esté redefiniendo, con los valores de sus ocho bytes de definición junto al mismo.

Las teclas de función sin el uso de la tecla SHIFT (f1, f3, f5, f7) controlan el movimiento del cursor dentro de esta ventana. La celda de debajo del cursor (correspondiente a un bit de uno de los ocho bytes de definición) se puede activar o desactivar con la tecla de función f2 más la tecla SHIFT. Cuando esto sucede se actualizan los ochos valores de definición e inmediatamente se puede ver cómo cambian todas las ocurrencias del carácter en la pantalla.

La pulsación de la tecla de función f4 con SHIFT permite que se reemplace el carácter de la ventana de edición por otro carácter. Los caracteres se describen mediante sus valores POKE (o código de pantalla) según están listados en el manual del usuario. Estos valores no son los mismos que los códigos CHR$ (aunque existe una correspondencia), pero su utilización resulta más conveniente en este caso porque las definiciones de caracteres están dispuestas en la memoria por el orden de estos códigos.

La tecla de función f6 más SHIFT permite escribir una copia del carácter que se está editando en la ventana de la derecha (o "de texto"), en la posición correspondiente a la del cursor de edición y en su tamaño real. Si, por ejemplo, el cursor estuviera en el rincón superior izquierdo de la ventana de edición y se estuviera editando el carácter "A", entonces se escribiría "A" en el rincón superior izquierdo de la ventana de textos cuando se pulsara f6.

Por último, pulsando la tecla del signo de exclamación se interrumpe el programa; CONT lo reiniciará. Cuando se abandona el programa, se puede digitar NEW y después cargar (LOAD) otro programa sin que se altere su juego de caracteres redefinidos. No obstante, ello supone ciertos problemas. En primer lugar, el tope de la memoria para el usuario se ha rebajado, de modo que sólo hay 12 Kbytes disponibles para el nuevo programa. En segundo lugar, al apagar la máquina se destruye el nuevo juego. En un futuro capítulo analizaremos ambos problemas. Mientras tanto, apunte las definiciones de sus nuevos caracteres redefinidos y utilícelos, si fuera necesario, como se muestra en el programa de la página 713.



Sin adorno
En la ventana izquierda, o de edición, está la versión redefinida del carácter número 55: el carácter "7". Se le ha quitado el rabo de la izquierda y se le ha agregado la barra cruzada continental. En la ventana derecha, o de texto, hay cuatro copias del siete. Los cambios son visibles en estos sietes y en los de alrededor de las ventanas, pero el siete del recuadro de estado conserva su definición original

Pinzas
Los sprites son tan sólo múltiples caracteres definidos por el usuario. Se han redefinido nueve caracteres y colocado juntos en la ventana de textos. El símbolo de tanto por ciento es, en su forma redefinida, la sección central derecha del cangrejo

Ian McKinnell

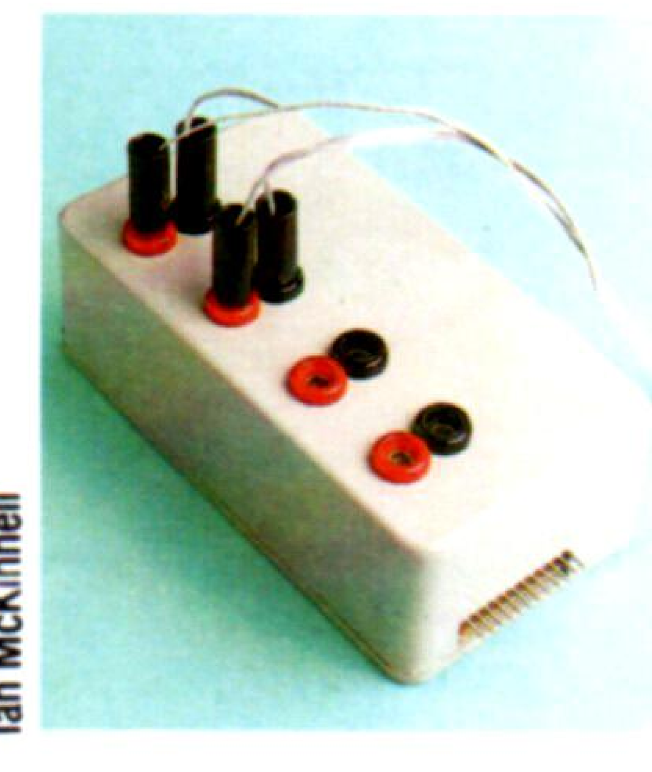


Definiendo caracteres

```
19 REM*************** C64 ****************
20 REM*  GEN. DE CAR. DEF. POR EL USUARIO  *
21 REM**********************************
50 POKE 52,56:POKE 56,56:CLR
60 PRINT CHR$(147)"POR FAVOR ESPERE 22 SEG"
70 GOSUB 61000: REM COPIAR JUEGO CAR.
100 GOSUB 1000: REM INICIALIZAR
120 FOR CT=0 TO 1 STEP 0
140 GOSUB 2500: REM INPUT
160 GOSUB 3000: REM VALIDAR
180 ON PT GOSUB 3500,4000,4500,7000
200 NEXT CT
900 END
999 REM**********************************
1000 REM*           INICIALIZAR           *
1001 REM**********************************
1020 DIM BD(8,8),C$(2,2),OF(2,7),CX(4)
1040 SH$=CHR$(19):SC$=CHR$(147):R$=CHR$(18):N$=
     CHR$(146):CD$=CHR$(17)
1060 P$=CD$+CD$+CD$:P$=P$+P$+P$+P$:P$=
     P$+P$:P$=SH$+P$
1080 C1$=CHR$(144):C2$=CHR$(5)
1200 REM ------ INICIALIZAR PANTALLA --------
1210 S0=1024:PRINT SC$C1$
1220 R0=4:C0=3:RL=8:CL=8:OF=16
1230 Z$="CARACTERES DEFINIDOS POR EL USUARIO"
1240 C=4:GOSUB2100:R=24:GOSUB2100
1250 L$="  76543210  ":S$="        "
1260 Z$=L$:R=R0:C=C0:GOSUB 2100
1270 C=C0-1:FOR R=R0+1 TO R0+8
1280 Z$=STR$(R-R0-1):Z$=Z$+S$+Z$
1290 GOSUB2100:C=C+OF:GOSUB2100
1300 C=C-OF:NEXT R
1310 C=C0:Z$=L$:GOSUB2100
1320 L$="  01234567  "
1330 Z$=L$:R=R0:C=C0+OF:GOSUB 2100
1350 C=C0+OF:R=R0+CL+1 :GOSUB2100
1370 PRINT C2$
1400 C$(1,1)=R$+"  "+N$:C$(1,2)="  "
1410 C$(2,1)=R$+"*"+N$:C$(2,2)="*"
1420 REM ________ DESPLAZ. CURSOR__________
1440 DATA 0,-1,+1,0,-1,0,0,+1
1460 FOR K=1 TO 2:FOR L=1 TO 4:
1480 READ OF(K,L):NEXT L,K
1500 REM ______ CONVERSION CARACS. ________
1520 DATA 64,0,32,64
1540 FOR K=0 TO 3:READ CX(K):NEXT K
1620 RP=1:CP=1:CN=1:GOSUB 6000
1990 RETURN
1999 REM**********************************
2000 REM*        PONER CRSR @ R, C         *
2001 REM**********************************
2050 PRINT LEFT$(P$,R+1)TAB(C);
2070 RETURN
2099 REM**********************************
2100 REM*          PRINT Z$ @ R,C          *
2101 REM**********************************
2150 PRINT LEFT$(P$,R+1)TAB(C)Z$;
2170 RETURN
2499 REM**********************************
2500 REM*     INTERMIT. CRSR @ RP,CP       *
2501 REM**********************************
2520 CF=1+BD(RP,CP):R=RP+R0:C=CP+C0
2540 FOR LP=0 TO 1 STEP 0
2560 FOR CS=1 TO 2:DE=10:GOSUB 2800
2580 GET GT$
2600 IF GT$ <> "" THEN LP=1:CS=2
2620 Z$=C$(CF,CS):GOSUB 2100
2640 DE=10: GOSUB 2800
2660 NEXT CS,LP:RETURN
2799 REM**********************************
2800 REM*        DEMORA PARA DE           *
2801 REM**********************************
2820 FOR NN=1 TO DE:NEXT:RETURN
2999 REM**********************************
3000 REM*       VALIDAR ENTRADA           *
3001 REM**********************************
3020 IF GT$="!" THEN R=18:C=0:GOSUB2000:STOP
3040 GT=ASC(GT$)-132:F=2*INT(GT/2)
3060 IF (GT<1)OR(GT>8) THEN PT=0:RETURN
3080 IF GT<5 THEN PT=1:RETURN
3100 PT=GT-3
3490 RETURN
3499 REM **********************************
3500 REM*        MOVER EL CURSOR          *
3501 REM **********************************
3520 NY=RP+OF(2,GT):NX=CP+OF(1,GT)
3540 IF (NY<1)OR(NY>RL) THEN RETURN
3560 IF (NX<1)OR(NX>CL) THEN RETURN
3580 RP=NY:CP=NX
3620 RETURN
3999 REM **********************************
4000 REM*       ACTIVAR UNA CELDA         *
4001 REM **********************************
4020 TG=1-BD(RP,CP):Z$=C$(1+TG,2)
4040 R=R0+RP:C=C0+CP:GOSUB2100
4060 BD(RP,CP)=TG:MP=NCGEN+CN*8-1
4120 PE=PEEK(MP+RP)
4140 PE=PE+(TG*2-1)*(2^(8-CP))
4160 POKE (MP+RP),PE:GOSUB6500:RETURN
4499 REM **********************************
4500 REM*    DEFINIR NUEVO CARACTER       *
4501 REM **********************************
4520 FOR K=1 TO 1
4540 Z$=R$+"      CAMBIAR CARAC.      "
4550 R=14:C=7:GOSUB2100
4560 Z$="    NUEVO NUMERO    "
4570 R=15:GOSUB2100
4580 C=19:GOSUB2000:INPUT A$
4600 CN=VAL(A$):PRINT C2$
4620 IF (CN<0)OR(CN>127) THEN K=0
4640 NEXT K:GOSUB 6000
4660 Z$="                    ":C=7
4670 R=14:GOSUB2100:R=15:GOSUB2100
4680 RETURN
5999 REM **********************************
6000 REM*       VISUALIZAR CARAC.         *
6001 REM **********************************
6020 MP=NCGEN+CN*8-1
6040 FOR RP=1 TO 8:PE=PEEK(MP+RP):Z$=""
6060 FOR CP=8 TO 1 STEP -1:N=INT(PE/2)
6080 Q=PE-2*N:BD(RP,CP)=Q:PE=N
6100 Z$=C$(Q+1,2)+Z$:NEXT CP
6120 R=R0+RP:C=C0+1:GOSUB2100
6130 NEXT RP
6140 X$=CHR$(CN+CX(INT(CN/32)))
6150 Z$=R$+"CARACTER":R=1:C=11
6160 GOSUB2100:R=R+2:GOSUB2100
6170 Z$="  "+X$+"=  "
6180 R=R-1:GOSUB2100
6190 C=C+4:Z$=STR$(CN)+N$:GOSUB2100
6220 RP=1:CP=1
6490 GOSUB6500:RETURN
6499 REM **********************************
6500 REM*       VISUALIZAR BYTES          *
6501 REM **********************************
6540 C=C0+CL+2:FOR R=R0+1 TO R0+8
6560 Z$=STR$(PEEK(MP+R-R0))+"  "
6580 GOSUB2100:NEXT:RETURN
6999 REM **********************************
7000 REM*       COLOCAR UN CARAC.         *
7001 REM **********************************
7020 Z$=X$:C=C+OF:GOSUB2100:RETURN
60999 REM **********************************
61000 REM*     REUBICAR CARAC. GEN.        *
61001 REM **********************************
61100 CGEN=53248:NCGEN=14336
61120 ITRPT=56334:IOPT=1:PO=53272
61125 RETURN
61150 POKE IT,PEEK(IT)AND254
61200 POKE IO,PEEK(IO)AND251
61250 FOR J=0 TO 2047
61300 POKE (NCGEN+J),PEEK(CGEN+J)
61350 NEXT J
61400 POKEIO,PEEK(IO)OR4
61450 POKE IT,PEEK(IT)OR1
61500 POKE PO,(PEEK(PO)AND240)OR14
61990 RETURN
```

Encender y apagar

Continuamos explicándole cómo construir un dispositivo interface que permita a su ordenador controlar pequeños aparatos caseros



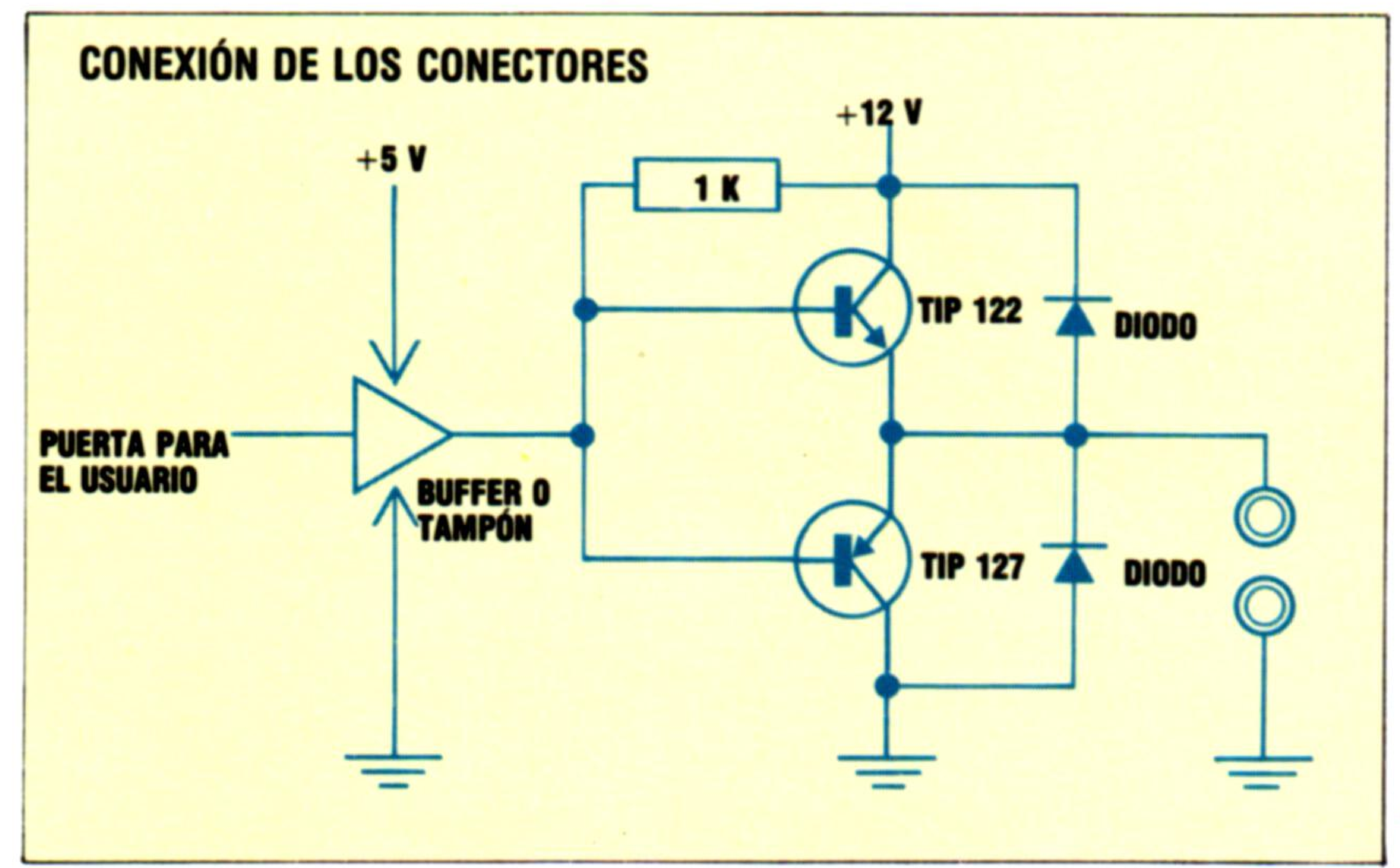
Ian McKinnell

La caja
Ésta es la caja terminada, mostrando el enchufe minicon y los cables de entrada-salida. Se debe tener cuidado al cortar la placa y la ranura de la caja de modo que la placa no se mueva en la caja cuando se inserta el enchufe en el conector

Esta interface posibilitará que su ordenador personal controle dispositivos de poco voltaje que consumen poca corriente. La salida de cada uno de los cuatro bits de la puerta para el usuario del ordenador es tamponada por el mismo chip de buffer que utilizamos en la construcción de una caja buffer en un capítulo anterior de este apartado (véase p. 1003). La salida del mismo se utiliza para conmutar los transistores, que son capaces de controlar voltajes y potencias superiores de los que puede manipular el buffer. Los dos diodos de la salida protegen a los transistores de corrientes inversas que pueden crear algunas cargas inductoras, como relés y motores.

La interface se puede utilizar como controlador de motores bidireccionales. Para conectar o desconectar uno de esos motores basta con colocarlo entre el conector de salida y el conector a tierra. Cuando sale un uno desde el ordenador, el motor arranca. Una salida cero apagará el motor.

La conexión del motor entre dos de las salidas de la interface, sin embargo, permitirá controlar también la dirección del movimiento del motor. Si el ordenador envía las mismas salidas (ambas ceros o ambas unos), entonces en ambas salidas de la interface aparecerá el mismo voltaje y a través del motor no fluirá ninguna corriente. Un uno en una salida y un cero en la otra darán una diferencia en voltajes que hará que el motor gire en una dirección. Invirtiendo la lógica, la corriente fluirá (y el motor girará) en la dirección contraria.



Liz Dixon

Construcción de la interface

En primer lugar, construya la caja en la cual se alojará la interface. La placa de circuitos encajará exactamente en la caja especificada para el proyecto de la caja buffer. La caja debe estar perforada para aceptar los conectores y el enchufe del bus (y el conector minicon si así se requiriera).

Después de haber fijado los conectores en la caja tenemos que hacer las conexiones entre ellos. Utilizando un trozo de cable estañado, conecte todos los conectores a tierra (negros) juntos; tome un trozo de 15 cm de cable plano de nueve vías y fije una hebra al cable estañado que conecta los conectores a tierra. Fije las otras ocho hebras, de a dos, a cada uno de los cuatro conectores de salida (rojos).

Ahora corte la veroboard a la medida correcta (45 agujeros por 16 franjas) y quite la media fila de agujeros de uno de los extremos. Conserve este retal de la placa, porque se utilizará para la construcción de la otra interface. Ahora efectúe los cortes de pistas tal como se ilustra en la fotografía A. Suelde primero los componentes pasivos: el conector del chip, el conector del bus y los enlaces de cables. Si desea instalar el conector de ampliación del bus, instálelo ahora, pero deje para el final el cable que conecta al mismo con el enchufe del bus. A continuación suelde las resistencias, seguidas de los diodos. Compruebe y asegúrese de que los mismos quedan instalados en la posición correcta. Seguidamente suelde los ocho transistores. Por último, suelde las conexiones a los conectores tal como se indica en la fotografía B e instale el chip en su lugar. Ahora puede ensamblar la placa en la caja y la interface ya está lista para utilizar.

Lista de componentes

Cantidad	Artículo
1	Veroboard 50 agujeros×24 líneas
4	Resistencia 1 K-ohm
8	Diodo 1N4001
4	Transistor TIP 122
4	Transistor TIP 127
1	7407
1	Conector DIL 14 patillas
4	Conector 4 mm rojo
4	Conector 4 mm negro
4	Enchufe 4 mm rojo
4	Enchufe 4 mm negro
1	Enchufe minicon 12 vías
1	Caja "2004" 120×65×40 mm
1	Conector minicon 12 vías*

* El último artículo es opcional. Amplía el bus del sistema más allá de esta interface, de modo que se puedan enchufar simultáneamente otras.
Debe contar con un poco de cable estañado y de cable plano que posiblemente le hayan sobrado del proyecto anterior.



Pistas
Tenga especial atención con los cortes de las pistas en esta placa, en particular los cortes individuales que separan los transistores

A

B

Componentes
Los TIP 122 están en el extremo derecho de la placa y los cuatro diodos entre ellos están conectados con el extremo negro a la pista del borde y con el extremo plateado a la pista siguiente. La línea de cable plano desde los conectores negros está soldada al extremo izquierdo de esta pista. Los otros cuatro diodos están conectados en el sentido opuesto entre los TIP 127. A uno de los lados de la placa se puede ver claramente el enchufe minicon

Los bucles, otra vez

Analicemos otra forma de establecer iteraciones en el código, similar en muchos aspectos al uso de las bifurcaciones

En anteriores capítulos, hemos comprobado cómo puede efectuarse una misma parte de un programa. Esto se consigue con el uso de las bifurcaciones, ya sean condicionales o incondicionales. Pues bien, existe una forma similar de establecer iteraciones en el código mediante las técnicas reconocidas por el BASIC como FOR...NEXT (general para todas las versiones) y WHILE...WEND (propia del tipo Microsoft y sus derivados), basadas en los mismos principios pero con claras diferencias. Establezcamos los puntos comunes entre un bucle tratado con FOR...NEXT y otro en la forma tradicional. El ejemplo en ambos casos será el mismo: la impresión de los números pares comprendidos entre 2 y 50. La figura 1 muestra el ordinograma junto con la transcripción del mismo a BASIC, tal como solemos hacerlo. Los componentes, tanto en una como en otra solución, son similares: un valor inicial (2) de la variable, y otro valor (50), final, que marca el tope que debe alcanzar dicha variable y, en este caso, asimismo el límite superior a visualizar. Por último, un valor de incremento (2), factor que va incrementando la variable hasta alcanzar el valor final.

En la línea 50 de la figura 1 se establece la comparación de la variable con el valor final para adoptar una decisión ante sus dos posibles salidas: final de proceso o continuación-iteración.

En la figura 2 puede observarse la existencia de los mencionados elementos, salvo el de la decisión (IF), ya que el ciclo FOR...NEXT lleva un contador interno que marca cuándo debe repetir el bucle o bien seguir en secuencia a la próxima instrucción. Así, y a modo de adaptación libre, se podría comentar el programa de la figura 2 diciendo: escribir repetidamente el contenido de la variable A empezando con un valor inicial 2, en incrementos de 2 unidades hasta alcanzar el valor 50.

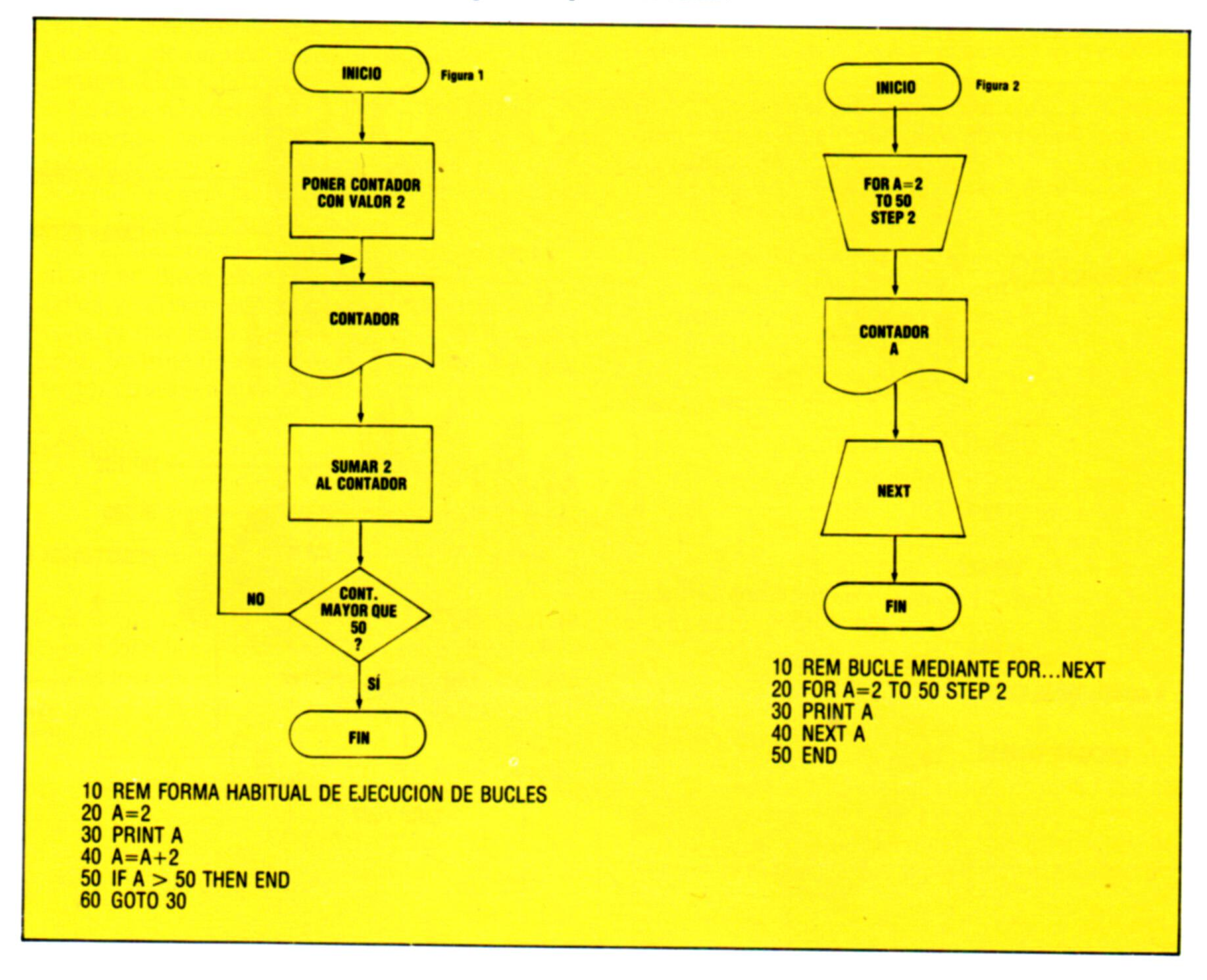
La representación del FOR...NEXT, WHILE...WEND en diagramación utiliza dos símbolos complementarios:

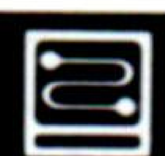
para apertura, frase inicio (FOR-WHILE)

y

para cierre de ciclo (NEXT-WEND)

También es posible (en caso de FOR) la especificación del incremento (STEP) de la variable de control del bucle.





Simple aritmética

Ya podemos examinar algunos programas con instrucciones del 6809 que realizan operaciones sencillas de aritmética

A este nivel del curso va siendo hora de juntar varias instrucciones en un solo programa que sea operativo, aunque para ello necesitamos algunas nuevas que todavía no conocemos junto con las maneras de representar datos. Comenzaremos por delinear un programa sencillo que sirva para convertir un número escrito en BCD (*Binary Coded Decimal*: decimal codificado en binario) en su equivalente binario.

Un número decimal codificado en binario es una manera muy útil de representar los números decimales en forma binaria cuando trabajamos con procesadores de ocho bits. Mediante esta representación, cada dígito en decimal se convierte en su equivalente binario. Por ejemplo, el número decimal 69 equivale en BCD a %01101001: los cuatro primeros bits, empezando por la derecha (1001), representan el 9 en binario, como sabe el lector, y los cuatro restantes (0110) son el 6. Nótese que si tomáramos %01101001 como un todo en binario su equivalente decimal no es 69 sino 105.

Nuestro programa convertidor empleará, entre otras, las siguientes instrucciones:

- LSR (*Logical Shift Right*: desplazamiento lógico a la derecha): Desplaza cada uno de los bits del operando un lugar a la derecha. El bit que ocupa el lugar más extremo a la derecha se separa y se coloca en el registro de código de condición del procesador como bit de arrastre. El lugar "vacío" que obviamente queda en el extremo izquierdo del operando se rellena con un cero.

- AND: Realiza la operación lógica AND sobre cada bit de un registro con los bits correspondientes del operando, quedando el resultado en el registro. Es una instrucción muy utilizada con el fin de "enmascarar" determinados bits: si un registro posee un bit puesto en 1, aplicándole AND con otro bit prevalecerá este último bit en el registro (recuérdese, 1 AND 0=0 y 1 AND 1=1). En el caso de que el bit del registro contenga un 0, el resultado será siempre 0. Por ejemplo si operamos un valor como %00001111 contenido en algún registro con el valor correspondiente a alguna posición de memoria mediante la operación AND de seguro obtendremos al menos para los cuatro bits más a la izquierda esos mismos ceros. Así:

```
%00001111   valor del registro
%10110110   valor de la posición de memoria AND
%00000110   resultado que queda en el registro
```

- MUL: Multiplica los contenidos de los registros A y B, llevando el resultado al registro D (que, como se dijo, está formado por la unión de los registros A y B y, por tanto, es de 16 bits). Hay muy pocos procesadores de ocho bits capaces de aceptar la multiplicación como un opcode.

- SWI (*SoftWare Interrupt*: interrupción por software): Es el modo más aconsejable de concluir un programa en código máquina y devolver el control al sistema operativo. Examinaremos con mayor detalle esta instrucción más adelante.

He aquí el programa "De BCD a binario":

1) Especificación de un valor para el contador de posiciones:

```
ORG $1000
```

2) Almacenamiento del número 58 en BCD dentro de la posición BCDNUM al tiempo que se reserva un byte en BINNUM:

```
BCDNUM  FCB   %01011000
BINNUM  RMB   1
```

3) Carga del número 58 en BCD dentro del registro A enmascarando el dígito inferior, y almacenamiento de éste en BINNUM:

```
START LDA   BCDNUM
      ANDA  #%00001111
      STA   BINNUM
```

4) Carga del 58 en BCD en el acumulador A y desplazamiento del dígito superior (o sea, los cuatro bits más a la izquierda) hacia la derecha:

```
SHIFT LSRA
      LSRA
      LSRA
      LSRA
```

5) Carga del número 10 (en decimal) en el registro B y multiplicación de éste por el contenido de A:

```
MULT   LDB #10
       MUL
```

6) El resultado será de 16 bits y queda en el registro D, pero como el resultado no puede ser mayor que 90 (10×9=90), sólo se necesita el byte inferior de D. Este byte se encuentra en el registro B, por eso el paso siguiente será sumar el contenido de B a BINNUM y almacenar el resultado:

```
ADDIT   ADDB BINNUM
        STB  BINNUM
```

7) Tenemos, pues, el número en BCD almacenado en BCDNUM y su equivalente binario almacenado en BINNUM. Devolveremos (*return*) el control al sistema operativo y finalizaremos (*end*) el programa:

```
RETURN  SWI
END
```

El complemento a dos

Hasta ahora hemos descrito programas que sólo realizaban sencillas operaciones aritméticas, y por este camino continuaremos durante algún trecho

Decimal	Binario
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

más. Nos interesa ahora estudiar el problema del *signo*, o sea los números negativos y positivos.

El método más conocido de representación de los números negativos dentro de una posición de memoria o de un registro es denominado *complemento a dos*. Es fácil obtener dicho complemento: se invierten todos los dígitos (los ceros se cambian en unos, y viceversa) y después se le suma un uno al número así obtenido. Por ejemplo, el complemento a dos de 0101 se obtiene así: 1010+1=1011.

Pero ¿cuál es su utilización en las operaciones matemáticas con números negativos? Ante todo, consideremos cuántos números se pueden representar: un registro de ocho bits sólo acepta 256 diferentes combinaciones de bits (2^8), combinaciones que podríamos distribuir así: los 127 primeros números serán positivos, los 128 restantes negativos, o sea, desde el −128 hasta el +127 (nótese que con un registro de 16 bits iríamos desde el −32768 hasta el +32767), pues admite 2^{16}=65536 números distintos). En la tabla adjunta mostramos la representación de valores desde el −7 hasta el +7, empleando sólo cuatro bits.

Si observa usted dicha tabla, notará que todos los números negativos tienen el MSB (*Most Significant Bit*: el bit más significativo, o sea, el situado más a la izquierda) puesto en 1. Igualmente para los positivos el MSB está puesto a 0.

Podemos entonces definir algunas propiedades básicas relacionadas con el complemento a dos y la aritmética de los signos:

- El complemento a dos de un número negativo nos da su positivo, y al revés.
- El bit más significativo (MSB) está a 0 para los números positivos, y a 1 para los negativos. Ésta es la contraseña por la que reconoceremos fácilmente si un número es positivo o negativo.
- El complemento a dos del número cero es cero (sume usted la unidad a 1111, p. ej.).
- La suma y la resta pueden realizarse del mismo modo, teniendo el resultado el signo correcto.

Puede que a usted le tiente probar esta última propiedad en algunos ejemplos de sumas y restas. Inténtelo. Desgraciadamente la multiplicación resulta algo más difícil cuando se usan números negativos. La instrucción MUL que utilizamos en el programa anterior para convertir BCD en binario suponía que los registros A y B contenían valores sin signo. Si deseamos multiplicar dos números con sus respectivos signos + o − (o sea, 0 o 1 en el MSB) debemos recurrir a un programa.

Cualquier lector con alguna práctica de programación sabe lo limitados que resultan los programas "lineales" hasta aquí empleados. Sólo podemos conseguir algo más útil si nos servimos de alguna de las formas básicas de estructura de control:

- Selección: en la que escogemos entre dos alternativas de acción (semejante al IF del BASIC).
- Repetición: por la que repetimos una secuencia de operaciones:

1) mientras (*while*) se cumplan ciertas condiciones (la estructura WHILE...WEND)
2) hasta que (*until*) se cumplan ciertas condiciones (la estructura REPEAT...UNTIL); o bien
3) un cierto número de veces (FOR...NEXT).

Todas estas estructuras dependen de la habilidad para verificar si una condición es verdadera o falsa, siendo la condición más común el que una variable tenga o no un determinado valor. En assembly necesitamos utilizar estas estructuras, y deberemos ser capaces de verificar los valores de los registros. Por lo general se pueden verificar directamente según dos posibilidades (si un valor es cero o no, si es positivo o negativo). Con otras instrucciones adicionales podemos, no obstante, realizar otros tipos de verificaciones.

El registro de código de condición

Lo anterior se consigue con el empleo de este registro de código de condición (CC), que ya se mencionó de pasada anteriormente (véase p. 1018). Se trata de un registro de ocho bits, pero de él, al contrario de los restantes registros, no nos interesa el valor que almacena. Nos interesa más bien el estado (1 o 0) de cada uno de sus ocho bits. Cinco de los ocho bits sirven para expresar las condiciones que hemos venido analizando hasta aquí, los otros tres se encargan del tratamiento de las interrupciones (que examinaremos más adelante). Uno de los cinco, el H (el flag de arrastre *mitad*; en inglés, *Half*), casi es exclusivo de las operaciones aritméticas en BCD, y por el momento no nos va a preocupar. Los cuatro restantes, que sí nos importan a este nivel, son:

- C: Flag de arrastre (inglés: *Carry*) que sirve para retener el bit de arrastre (o de sustracción, en caso de la resta) tomado del bit más significativo tras una operación aritmética. Tiene también una función muy útil en caso de que queramos desplazar el contenido de un acumulador un bit nada más; varias de las operaciones de desplazamiento colocan el bit que se desecha en C. Este bit puede servir, por ejemplo, para verificar si el número es par o impar simplemente llevando el bit menos significativo a C y comprobando su valor. Se trata del bit 0 (el bit menos significativo) en el CC.
- V: Flag de desbordamiento (inglés: *oVerflow*), puesto a 1 cuando el resultado de una operación aritmética no cabe, por excesivamente grande, dentro del registro que debería contenerlo. Es el bit 1 en el CC.
- Z: Es el flag cero (inglés: *Zero*), puesto a 1 cuando el contenido de un registro es cero. Se trata del bit 2 en el CC.
- N: Flag de los negativos. Es una copia del bit más significativo (el bit del signo) del contenido de un registro; o sea, se pone a 1 si el número es negativo. Se trata del bit 3 en el CC.

Uno de los aspectos más difíciles del lenguaje assembly a la hora de programar, es atender el estado de los flags. No todas las instrucciones activarán los flags, y algunos de éstos se activan según el contenido del acumulador mientras otros pueden también sufrir cambios según otros registros. El procedimiento más seguro es verificar sólo los valores contenidos en un acumulador, y realizarlo en el momento en que aparece el valor requerido, ya que es difícil asegurar que los flags no cambien con la intervención de cualquier otra instrucción.

Los flags son verificados por medio de instrucciones de bifurcación, las equivalentes a bajo nivel, del GOTO en BASIC. El 6809 emplea bifurcaciones relativas (más que absolutas) de modo casi exclusivo. La diferencia está en que una bifurcación relativa transfiere el control un cierto número de bytes hacia adelante (o hacia atrás), mientras que una bifurcación absoluta transfiere el control a una dirección absoluta especificada. Pero el efecto es el mismo. Hay que distinguir entre bifurcaciones *cortas*, cuando la amplitud se puede expresar en un solo byte (del −128 al 127), y bifurcaciones *largas*, que pueden dirigir a cualquier punto de la memoria. Continuaremos usando tan sólo bifurcaciones cortas.

El 6809 tiene un extenso conjunto de instrucciones de bifurcación, que iremos presentando a medida que las vayamos necesitando. Los ejemplos que siguen ilustran las instrucciones empleadas para verificar y comparar los valores retenidos en los acumuladores. Ilustran igualmente el uso de las instrucciones de bifurcación en los procedimientos de selección y repetición.

- ANDCC: No es posible cargar valores directamente dentro del registro de código de condición, pero es una buena práctica poner a cero todos los flags que va usted a necesitar antes de comenzar a usarlos. La forma más fácil de hacer esto es servirse de la instrucción ANDCC, que opera como la orden AND, empleando ceros como máscaras en las posiciones de bit que deseamos usar.
- SUB (SUB*tract*: resta): Resta el operando del contenido del acumulador, y afecta a los flags C, V, Z y N, según el resultado de la operación (incluso el flag H se activa si la resta es de ocho bits).
- CMP (CoMParación): Funciona igual que SUB, sólo que no cambia el contenido del registro. Afecta, como en SUB, a los flags C, V, Z, N (y H).
- BRA (*Inconditional* BRA*nch*: bifurcación incondicional): Parecida al GOTO del BASIC.
- BGT (B*ranch if* G*reater* T*han zero*: bifurcar si mayor que cero): Se comprueba con ella el signo de los números. La bifurcación tiene lugar si Z es cero (el número no es cero). En previsión de que el signo pueda ser incorrectamente interpretado si hubo desbordamiento, N y Z deben ser cero (directamente, no negativos) o bien N y V deben ser uno (negativo equivocado por causa del desbordamiento). Otras verificaciones similares para los números con signo son BGE, BLT y BLE ("bifurcar si mayor o igual", "si menor que" y "si menor o igual").
- BLO (B*ranch if* LO*wer than zero*: bifurcar si es menor que cero): Se trata de una verificación sin signo, pues no tiene sentido comprobar N para un número sin signo. La bifurcación se produce cuando el flag C se pone a 1, el cual indica la unidad a quitar en una resta. Verificaciones similares son BLS, BHI y BHS.
- He aquí un programa para encontrar el número más grande entre dos con signo y de ocho bits almacenados en $3000 y $3001. El resultado se colocará en $3002:

1) Etiquetaremos los números:

```
NUM1    EQU    $3000
NUM2    EQU    $3001
RES     EQU    $3002
        ORG    $1000
```

2) Iniciamos la codificación: debemos poner a cero todos los flags del código de condición y cargar el primer número. Éste se compara con el otro:

```
        ANDCC  %11110000
        LDA    NUM1
        CMPA   NUM2
```

3) Si el más grande es NUM1, el programa bifurcará a FIN, de lo contrario cargará el segundo número en el registro A. Cualquiera que sea el número que se halle en el registro cuando se pase a FIN, irá a almacenarse a RES, y el programa devolverá el control al sistema operativo, concluyendo (END), entonces, de la siguiente manera:

```
        BGT    FIN
        LDA    NUM2
FIN     STA    RES
        SWI
        END
```

Directivas originales
Los diferentes efectos que las directivas del ensamblador y las sentencias en assembly tienen sobre el contador de posiciones y sobre el contenido de la memoria pueden apreciarse en el siguiente ejemplo

Directivas originales

*----------DEMONSTRATION--------------

CAMPO DE ETIQUETA	CAMPO OPCODE	CAMPO OPERANDOS	CONTADOR POSICIONES	CONTENIDO MEMORIA	
RESET	EQU	$F100	$0400	??	Como no se puso ningún ORG, la dirección de la posición es la que el ensamblador tiene por defecto. La dirección no queda afectada por EQU, y el contenido de la memoria está todavía por definir
INDEX	EQU	16	$0400	??	
MASK1	EQU	%01101010	$0400	??	
	ORG	$1000	$1000	??	Sitúa la posición según se indica, pero el contenido de la memoria sigue indefinido
CR	FCB	16	$1000	$10	FCB hace que el operando quede almacenado en el byte direccionado por el contador de posiciones
MEMTOP	FDB	$7FFF	$1001	$7F	El FDB inicializa dos bytes
			$1002	$FF	
TABLE1	RMB	7	$1003	??	RMB reserva 7 bytes (de contenido indefinido) incrementando el contador de posiciones en ese número
			$1004	??	
			$1005	??	
			$1006	??	
			$1007	??	
			$1008	??	
			$1009	??	
ERRMSG	FCC	'ERROR	$100A	$45	Los códigos ASCII del operando string son colocados en la memoria gracias a la directiva FCC
			$100B	$52	
			$100C	$52	
			$100D	$4F	
			$100E	$52	
	CLRA		$100F	$4F	¡Por fin una operación en assembly! No hay operando alguno; disponemos sólo de un byte como opcode
	END		$100F	??	Otra directiva que no afecta al contador de posiciones

```
*-------SYMBOL TABLE-----------------
RESET   F100   INDEX  0010   MASK1   006A
CR      1000   MEMTOP 1001   TABLE1  1003
ERRMSG  100A
```

Así es como se almacenarán los símbolos utilizados en el programa dentro del espacio de trabajo del ensamblador para uso propio durante el assembly

Una nueva meta

Memotech, proveedor de accesorios y periféricos para el Sinclair ZX81, ya está produciendo su propia gama de micros: la serie MTX



Línea especial
Memotech empezó con productos especializados para el Sinclair ZX81, como el Memopack, que vemos en la fotografía, que le proporcionaba al ZX81 32 K adicionales de Ram

Memotech se creó a raíz del enorme interés del público por los primeros microordenadores de Sinclair Research. A pesar de la popularidad del ZX80 y del ZX81, pronto se hizo evidente que las máquinas estaban severamente limitadas por una carencia de memoria, y se creó un inmenso mercado para placas de memoria adicionales.

Los dos fundadores de esta empresa británica eran catedráticos de la Universidad de Oxford: Geoff Boyd disertaba sobre metalurgia en Wilson College, y Robert Branton enseñaba matemáticas en Christ Church. Los dos se encontraron por primera vez en una muestra de informática en la universidad, en 1981, y decidieron trabajar juntos en accesorios para el ZX81. Su primer producto fue una placa de ampliación de 16 Kbytes, que fue seguido por una serie completa de "Memopacks", incluyendo paquetes de RAM de 32 Kbytes y 64 Kbytes, el paquete para gráficos en alta resolución (HRG), un análisis de hoja electrónica (Memocalc), un procesador de textos (Memotext), interfaces Centronics y RS232 y un teclado.

La gama MTX se lanzó oficialmente en febrero de 1984 y la empresa afirma que desde entonces se han vendido alrededor de 25 000 máquinas. Al igual que el BBC Micro, el MTX viene en dos modelos: el MTX500, de 32 Kbytes, y el MTX512, de 64 Kbytes. Las máquinas utilizan un microprocesador Z80A y ofrecen 16 colores en modalidad de alta resolución (256×192 pixels). El BASIC MTX es similar al BASIC BBC. El ordenador también dispone de un ensamblador-desensamblador.

El ordenador también se puede ampliar para hacer uso del paquete para gráficos HRX de Memotech. Partiendo de un MTX500 sin ampliar, el usuario puede añadir unidades de disco y las tres placas controladoras de gráficos: una placa de control principal con un procesador de 86 bits, un "Frame Grabber" y un convertidor de A/D de tres canales. El sistema resultante es capaz de producir animaciones, composición de imágenes y diseño gráfico hasta una capacidad total de composición tipográfica.



Frutos del éxito
Después del éxito en productos de ampliación para el ZX81, Memotech avanzó hacia nuevos productos. Uno es una línea de periféricos para ordenadores, como la impresora matricial de 80 columnas, la DMX 80.
La última realización de Memotech es el MTX512, un pequeño microordenador personal y de oficina de 64 K. El MTX512 puede almacenar datos en cassette o en la unidad de disco flexible opcional que vemos en la fotografía

Cuando en 1982 Sinclair Research lanzó el Spectrum, Memotech decidió no producir ninguna gama de accesorios para la nueva máquina. En lugar de ello, utilizando la experiencia y la pericia obtenidas con la producción del hardware para el ZX81, Memotech optó por concentrar todos sus recursos en diseñar y construir sus propias máquinas. Tim Spencer, director de ventas y marketing de Memotech, explica: "Pensamos que el ZX81 no iba a durar mucho tiempo más, de modo que decidimos construir nuestro propio ordenador. Al fin y al cabo, teníamos la tecnología. Pero el ZX81 ha durado mucho tiempo más del que esperábamos, y nuestros paquetes se siguen vendiendo bien".

Memotech calcula que las ventas internacionales de sus Memopacks han sobrepasado las 250 000 unidades. Los paquetes, así como la gama de máquinas MTX, se fabrican en las oficinas centrales de la empresa, en Witney (Oxfordshire). En la actualidad el personal de la empresa sobrepasa los 100 empleados.

Preguntado sobre la filosofía de diseño de la gama MTX, Tim Spencer respondió: "Estamos dirigiéndonos al usuario personal más serio y al mercado de gestión. Las máquinas no están dirigidas al mercado de juegos aunque, por supuesto, con ellas uno puede practicar todos los juegos más conocidos".

Debido a que el MTX es capaz de ejecutar CP/M, puede aprovechar la gama de software disponible. Sin embargo, Memotech es consciente de una falta de software basado en cassette, que haría que la máquina le resultara más atractiva al usuario personal menos serio. En la actualidad sólo hay alrededor de 40 cassettes diferentes a la venta para el MTX, y la empresa está estimulando activamente el desarrollo de más programas. "Hemos hecho muchísimo durante estos últimos meses", nos comentaba Tim Spencer. "Tenemos estrechas relaciones con Continental Software, y PSS está escribiendo para nosotros." En el futuro cercano también habrá diversos paquetes educativos.

Editorial Delta, S.A.



10053

9 788485 822836